SIEMENS

SIMATIC

使用 STEP 7 V5.3 编程

手册

2004 年 1 月版
A5E00446499-01

安全指南

本手册包括了保证人身安全与保护本产品及连接的设备所应遵守的注意事项。这些注意事项在手册中均以下列符号加以突出，并根据危险等级标明如下：

危险
表示如果不采取适当的预防措施，将导致死亡、严重的人身伤害或财产损失。

警告
表示如果不采取适当的预防措施，可能导致死亡、严重的人身伤害或财产损失。

警告
表示如果不采取适当的预防措施，可能导致轻微的人身伤害。

警告
表示如果不采取适当的预防措施，可能导致财产损失。

须知
提醒您注意有关产品、产品使用的特别重要的信息，或者是文档的特定部分。

合格人员

只有**合格人员**才允许安装和操作该设备。合格人员是指被授权按照既定安全惯例和标准，对线路、设备和系统进行调试、接地和标记的人员。

正确使用

请注意如下事项：

警告
该设备及其部件只能用于产品目录或技术说明书中所描述的范畴，并且只能与由西门子公司认可或推荐的第三方厂商提供的设备或部件一起使用。

只有正确地运输、保管、设置和安装本产品，并且按照推荐的方式操作和维护，产品才能正常、安全地运行。

商标

SIMATIC®、SIMATIC HMI®和 SIMATIC NET®是 SIEMENS AG 的注册商标。

本文档中的其它一些标志也是注册商标，如果任何第三方出于个人目的而使用，都会侵犯商标所有者的权利。

免责声明

我们已检查过本手册中的内容与所描述的硬件和软件相符。由于差错在所难免，我们不能保证完全一致。我们会定期审查本手册中的内容，并在后续版本中进行必要的更正。欢迎提出改进意见。

Siemens AG
Bereich Automation and Drives
Geschaeftsgebiet Industrial Automation Systems
Postfach 4848, D- 90327 Nuernberg

前言

目的

该手册完整概述了如何使用 STEP 7 编程。它设计用于在安装和调试该软件时为您提供支持。它解释了如何创建程序并描述了用户程序的组件。

该手册旨在用于供使用 STEP 7 和 SIMATIC S7 自动化系统执行控制任务的人员使用。

建议通过手册“使用 STEP 7 V5.3 版本，使用入门”中的实例熟悉该软件的使用。这些实例简单介绍了“使用 STEP 7 编程”的主题。

基础知识要求

要了解本手册，需要具有自动化技术的常规知识。

此外，必须熟悉安装了 MS Windows 2000 Professional 或 MS Windows XP Professional 操作系统的计算机或 PC 类工具(例如，编程设备)。

手册应用范围

本手册适用于 STEP 7 编程软件包 5.3 版本。

可以在 service pack 上获得最新信息：

- 位于“readme.wri”文件中
- 位于已更新的 STEP 7 在线帮助中。

在线帮助中“新增内容”主题提供了极好的介绍并概述了最新的 STEP 7 特征。

STEP 7 文档数据包

该手册是文档数据包“STEP 7 基本信息”的一部分。

下表显示了 STEP 7 文档的总览：

文档	目的	订货号
STEP 7 基础信息 • STEP 7 V5.3，使用入门手册 • 使用 STEP 7 V5.3 编程 • 组态硬件和通讯连接，STEP 7 V5.3 版本 • 从 S5 到 S7，变频器手册	提供给技术人员的基础信息，描述了使用 STEP 7 和 S7-300/400 可编程控制器来实现控制任务的方法。	6ES7810-4CA07-8BW0
STEP 7 参考书目 • 用于 S7-300/400 的梯形图(LAD)/功能块图(FBD)/语句表(STL)手册 • S7-300/400 的标准功能及系统功能	提供了参考信息，并描述了编程语言 LAD、FBD、STL、标准功能以及系统功能，扩充了 STEP 7 基础信息的范围。	6ES7810-4CA07-8BW1

在线帮助	目的	订货号
STEP 7 帮助	以在线帮助的形式提供关于使用 STEP 7 编程和组态的基本信息。	STEP 7 标准软件中的一部分。
STL/LAD/FBD 帮助参考 SFB/SFC 帮助参考 组织块帮助参考	上下文相关参考信息。	STEP 7 标准软件中的一部分。

在线帮助

集成于软件中的在线帮助是对本手册的补充。提供在线帮助的目的是，在使用该软件时提供详细的支持。

该帮助系统通过一些界面集成于软件中：

- 在**帮助**菜单中有多个菜单命令可供选择：**目录**命令打开关于 Step 7 帮助信息的索引。
- **使用帮助**提供了关于使用在线帮助的详细指示。
- 上下文相关帮助提供关于当前语境的信息，例如，打开的对话框或激活的窗口。通过单击“帮助”按钮或按下 F1 键可以打开上下文相关的帮助。
- 状态栏提供另一种形式的上下文相关的帮助。当鼠标指针放在菜单命令上时，它显示每个菜单命令的一个简短解释。
- 当鼠标指针在图标上停留一小段时间时，还为工具栏中的每个图标显示一个简要解释。

如果喜欢按打印格式阅读在线帮助信息，则可以打印输出单个帮助主题，或整个在线帮助。

该手册、“使用 STEP 7 配置硬件”手册、“通过 CiR 在操作期间修改系统”手册以及“自动化系统 S7 400H - 容错系统”摘自关于 STEP 7 的 HTML 版的帮助信息。要获得详细的步骤，请参见 STEP 7 帮助。由于这些手册和在线帮助共用一个几乎完全相同的结构，因此很容易在这些手册和在线帮助之间进行切换。

可以在安装 STEP 7 后，通过 Windows 开始菜单查找电子版手册：**开始 > SIMATIC > 文档**。

更多支持

如果有任何技术问题，请联系西门子代表或代理商。

您可以在下列网页中查找联系人：

http://www.siemens.com/automation/partner

培训中心

西门子提供了很多培训教程，引导您熟悉 S7 自动化系统。请联系当地的培训中心，或位于德国纽伦堡(D 90327)的培训总部，以获取详细信息。

电话：　+49(911) 895-3200。

网址：　http://www.sitrain.com

A&D 技术支持

遍布世界各处，24 小时服务：



全球(纽伦堡) **技术支持** 每年 365 天，每天 24 小时 电话： +49(180) 5050-222 传真： +49(180) 5050-223 电子邮件： adsupport@siemens.com 格林威治标准时间： +1:00		
欧洲/非洲(纽伦堡) **授权** 当地时间： 周一至周五，8:00 - 5:00 PM 电话： +49(180) 5050-222 传真： +49(180) 5050-223 电子邮件： adsupport@siemens.com 格林威治标准时间： +1:00	美国(约翰逊城) **技术支持和授权** 当地时间： 周一至周五，8:00 - 5:00 PM 电话： +1(423) 262 2522 传真： +1(423) 262 2289 电子邮件： simatic.hotline@sea.siemens.com 格林威治标准时间： -5:00	亚洲/澳洲(北京) **技术支持和授权** 当地时间： 周一至周五，8:00 - 5:00 PM 电话： +86 10 64 75 75 75 传真： +86 10 64 74 74 74 电子邮件： adsupport.asia@siemens.com 格林威治标准时间： +8:00
SIMATIC 热线以及授权热线所使用的语言通常为德语和英语。		

Internet 服务和支持

除文档以外，还在 Internet 上在线提供了知识产权信息，网址如下：

http://www.siemens.com/automation/service&support

可在其中查找下列内容：

- 公司简讯，经常提供产品的最新信息。
- 相应文档资料，可通过“服务和支持”中的搜索功能查找。
- 论坛，世界各地的用户和专家可以在此交流经验。
- 当地自动化和驱动办事处。
- 在“服务”页面下提供了关于现场服务、维修、备件等信息。

目录

1 介绍该产品并安装该软件

1.1 STEP 7概述

什么是STEP 7？

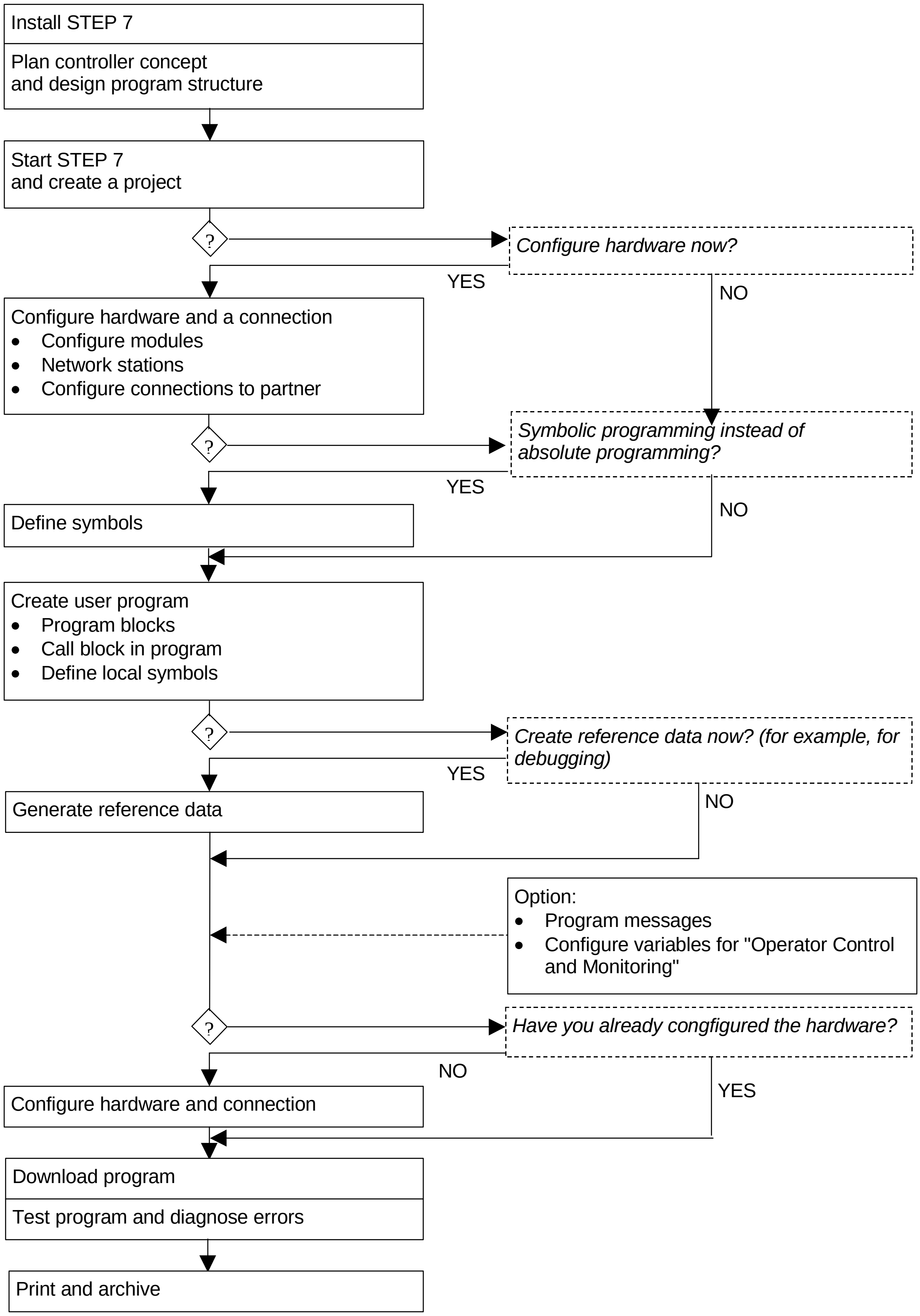
STEP 7 是一种用于对 SIMATIC 可编程逻辑控制器进行组态和编程的标准软件包。它是 SIMATIC 工业软件的一部分。STEP 7 标准软件包有下列各种版本：

- STEP 7 Micro/DOS 和 STEP 7 Micro/Win，用于 SIMATIC S7-200 上的简化单机应用程序。
- STEP 7，应用在 SIMATIC S7-300/S7-400、SIMATIC M7-300/M7-400 以及 SIMATIC C7 上，它具有更广泛的功能：
  - 可扩展到 SIMATIC 工业软件的软件产品中作为一个选件(参见 STEP 7 标准软件包的扩展使用)。
  - 功能模块和通讯处理器参数分配的时机
  - 强制模式与多值计算模式
  - 全局数据通讯
  - 使用通讯功能块进行的事件驱动数据传送
  - 组态连接

STEP 7 是本文档的主题，STEP 7 Micro 将在“STEP 7 Micro/DOS”文档中进行介绍。

基本任务

当您使用 STEP 7 创建一个自动化解决方案时，您将面对一系列的基本任务。下图给出了大多数项目都需要执行的任务，并将其分配给一个基本步骤。它指出了相关的参考章节，以使您方便地浏览手册，找到与任务相关的信息。



Install STEP 7
Plan controller concept
and design program structure
Start STEP 7
and create a project
?
Configure hardware now?
YES
NO
Configure hardware and a connection
• Configure modules
• Network stations
• Configure connections to partner
?
Symbolic programming instead of absolute programming?
YES
NO
Define symbols
Create user program
• Program blocks
• Call block in program
• Define local symbols
?
Create reference data now? (for example, for debugging)
YES
NO
Generate reference data
Option:
• Program messages
• Configure variables for "Operator Control and Monitoring"
?
Have you already congfigured the hardware?
NO
YES
Configure hardware and connection
Download program
Test program and diagnose errors
Print and archive

其它步骤

如上图所示，您有两个可供选择的方法：

- 您可首先组态硬件，然后对块进行编程。
- 然而，您也可首先对块进行编程，而不组态硬件。在保养和维护工作时，建议采用此方法，例如，将已编程的块集成到现有的项目中。

单个步骤的简短描述

- 装STEP 7和许可证密钥
  在第一次使用STEP 7时，对其进行安装，并将许可证密钥从软盘传送到硬盘(参见安装STEP 7和许可证)。
- 规划控制器
  在使用STEP 7进行工作之前，对自动化解决方案进行规划，将过程分解为单个的任务，并为其创建一个组态图(参见规划自动化项目的基本过程)。
- 设计程序结构
  使用STEP 7中可供利用的块，将控制器设计草图中所描述的任务转化为一个程序结构(参见用户程序中的块)。
- 启动STEP 7
  通过Windows用户接口启动STEP 7 (参见启动STEP 7)。
- 创建项目结构
  项目类似一个文件夹，所有的数据均可按照一种体系化的结构存储在其中，并随时可供使用。在项目创建完毕之后，所有其它的任务均将在该项目中执行(参见项目结构)。
- 组态一个站
  在对站进行组态时，您可指定您希望使用的可编程控制器；例如，SIMATIC 300、SIMATIC 400、SIMATIC S5 (参见插入站)。
- 组态硬件
  在对硬件进行组态时，您可在组态表中指定自动化解决方案要使用的模块以及用户程序用来对模块进行访问的地址。使用参数也可对模块的属性进行设置(参见硬件组态的基本步骤)。
- 组态网络和通讯连接
  通讯的基础是预先组态的网络。为此，您需要创建自动化网络所需要的子网、设置子网属性、以及设置已联网工作站的网络连接属性和某些通讯连接(参见子网的组态步骤)。
- 定义符号
  您可在符号表中定义局部符号或具有更多描述性名称的共享符号，以便代替用户程序中的绝对地址进行使用(参见创建符号表)。

- 创建程序
  使用一种可选编程语言创建一个与模块相链接或与模块无关的程序，并将其存储为块、源文件或图表(参见关于在STL源文件中进行编程的基本信息和STL源文件中关于编程的基本信息)。
- 仅适用于S7：生成并赋值引用数据
  您可充分利用这些引用数据，使得用户程序的调试和修改更容易(参见关于可用引用数据的概述)。
- 组态消息
  例如，通过其文本和属性，创建相关块的消息。使用传送程序，将所创建的消息组态数据传送给操作员接口系统数据库(例如，SIMATIC WinCC、SIMATIC ProTool)，参见组态消息。
- 组态操作员监控变量
  一旦在STEP 7中创建了操作员监控变量，就要为其分配所需要的属性。使用传送程序，将所创建的操作员监控变量传送到操作员接口系统WinCC的数据库(参见对用于操作员监控的变量进行组态)。
- 将程序下载给可编程控制器
  仅适用于 S7：在完成所有的组态、参数分配、以及编程任务之后，您可将整个用户程序或其中的单个块下载给可编程控制器(硬件解决方案的可编程模块)。(参见下载要求。) CPU 已经包含有操作系统。
  仅适用于 M7：从众多不同的操作系统中为您的自动化解决方案选择一个适合的操作系统，并将它独自或随用户程序一起传送给所需要的 M7 可编程控制系统的数据介质。
- 测试程序
  仅适用于 S7：为了进行测试，您可显示用户程序或 CPU 中的变量值，为变量分配数值，以及为您想要显示或修改的变量创建一个变量表(参见使用变量表进行测试的说明)。
  仅适用于 M7：使用高级语言调试工具对用户程序进行测试。
- 监视操作、诊断硬件
  通过显示关于模块的在线信息，确定模块故障的原因。借助于诊断缓冲区和堆栈内容，确定用户程序处理中的错误原因。也可检查用户程序是否可在特定的CPU上运行(参见诊断硬件和显示模块信息)。
- 归档设备
  在创建项目/设备之后，一件很有意义的事，就是为项目数据制作清楚的文档，从而使项目的编辑以及某些维修活动更容易(参见打印项目文档)。
  DOCPRO，用于创建和管理设备文档的一种可选工具，使您能够对项目数据进行结构化，将其转化为接线手册的形式，以及使用常见的格式进行打印。

特殊的主题

当创建一个自动化解决方案时，您可能要用到一些很有用处的特殊主题：

- 多值计算 - 多个 CPU 的同步操作(参见多值计算 - 多个 CPU 的同步操作)。
- 一个以上的用户在项目中进行工作(参见一个以上的用户编辑项目)。
- 使用 M7 系统进行工作(参见 M7 系统的步骤)。

1.2 STEP 7标准软件包

所使用的标准

集成在STEP 7中的SIMATIC编程语言符合EN 61131-3标准。该标准软件包符合面向图形和对象的Windows操作原则，在MS Windows 2000专业版(从现在起称为Windows 2000)以及MS Windows XP专业版(从现在起称为Windows XP)和MS Windows Server 2003操作系统中运行。

标准软件包的功能

标准软件在自动化任务创建过程的所有阶段都将给予支持，比如：

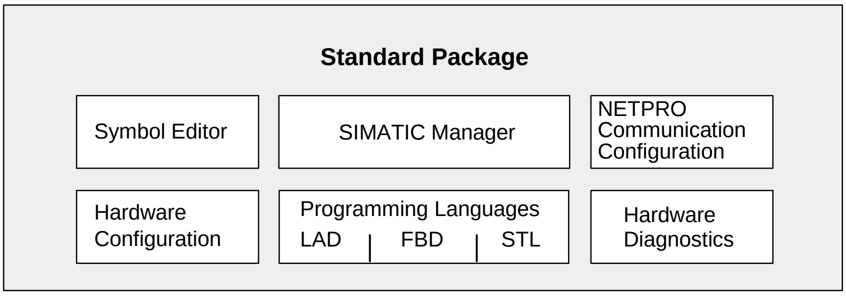
- 设置和管理项目
- 为硬件和通讯组态并分配参数
- 管理符号
- 创建程序，例如，用于 S7 可编程控制器
- 将程序下载到可编程控制器
- 测试自动化系统
- 诊断设备故障

STEP 7 软件用户界面的设计符合最先进的人类工程学，且易于入门。

STEP 7 软件产品文档提供有在线帮助和 PDF 格式电子手册中的所有在线信息。

STEP 7中的应用程序

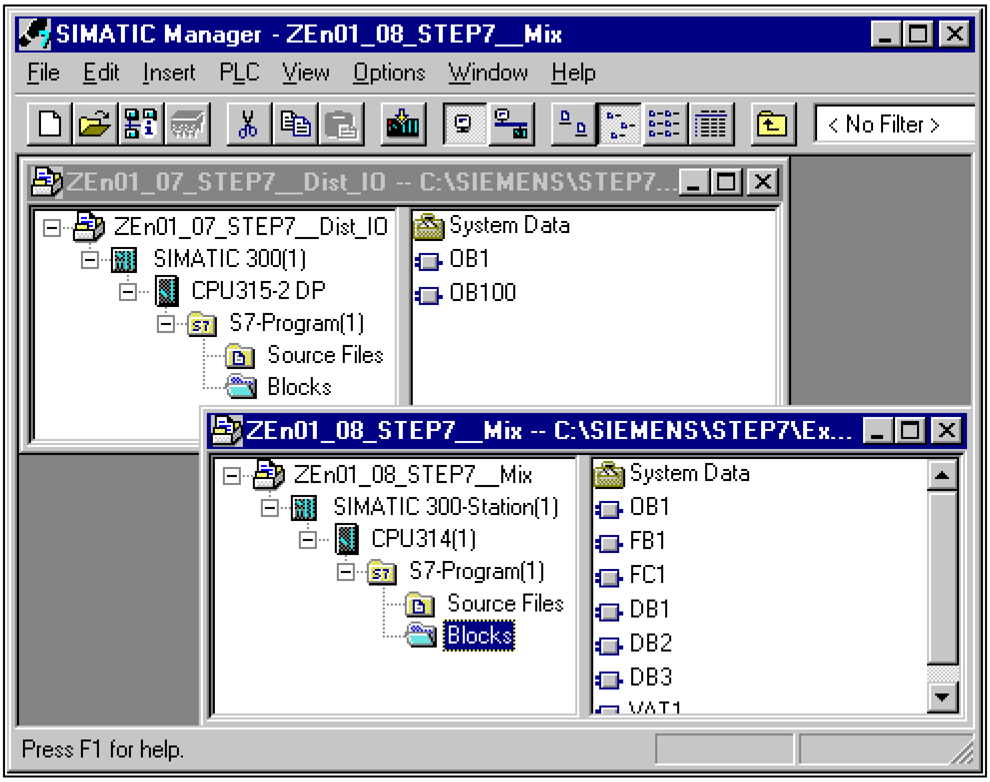
STEP 7 标准软件包中包含有一系列应用程序(工具)：



没有必要单独打开这些工具；在选择相应功能或打开对象时，将会自动启动这些工具。

SIMATIC管理器

SIMATIC 管理器管理一个自动化项目中的所有数据，而无论其设计用于何种类型的可编程控制系统(S7/M7/C7)。编辑数据所需的工具由 SIMATIC 管理器自动启动。



符号编辑器

通过符号编辑器，可以管理所有共享符号。符号编辑器提供下列功能：

- 给过程信号(输入/输出)、位存储器以及块设置符号名称和注释
- 排序功能
- 从其它 Windows 程序中导入/导出到其它 Windows 程序

所有其它工具都可使用该工具创建的符号表。因此，符号属性的任何变化都可被所有工具自动识别。

硬件诊断

这些功能可以概览可编程控制器的状态。概览可显示符号来指示各个模块是否发生故障。双击故障模块可显示关于故障的详细信息。该信息范围取决于每个模块：

- 显示模块的常规信息(例如，订货号、版本、名称)以及模块状态(例如，故障状态)
- I/O 和 DP 从站的模块故障(例如，通道故障)
- 显示来自诊断缓冲区的消息

对于 CPU，则显示下列附加信息：

- 处理用户程序期间发生故障的原因
- 显示周期持续时间(最长、最短以及最后一个周期)
- MPI 通讯概率和负载
- 显示性能数据(输入/输出、位存储器、计数器、计时器和块的可能数目)

编程语言

S7-300 和 S7-400 的编程语言梯形图、语句表和功能块图是标准软件包的一个重要组成部分。

- 梯形图(或 LAD)是 STEP 7 编程语言的图形表示。其指令语法与传递梯形图相似：梯形图允许在能流过各种触点、复杂元件和输出线圈时，跟踪母线之间的能流量。
- 语句表(或 STL)是 STEP 7 编程语言的文本表示，与机器代码相似。如果用语句表书写程序，则每条指令都与 CPU 执行程序的步骤相对应。为便于编程，语句表已经扩展包括一些高级语言结构(如结构化数据访问和块参数)。

- 功能块图(FBD)是 STEP 7 编程语言的图形表示，使用布尔代数惯用的逻辑框表示逻辑功能。复杂功能(如算术功能)可直接结合逻辑框表示。

其它编程语言则作为选件包提供。

硬件配置

使用该工具可对自动化项目的硬件进行配置并分配参数。符号编辑器提供下列功能：

- 要组态可编程控制器，可从电子目录中选择机架，然后在机架所要求的插槽中排列所选模块。
- 组态分布式 I/O 与组态集中式 I/O 相同。也支持具有通道式 I/O。
- 分配 CPU 参数期间，可以设置属性，如启动特性和通过菜单导航的扫描周期监控。支持多值计算。输入数据存储在系统数据块中。
- 分配模块参数期间，通过对话框设置所有可设定的参数。不需要通过 DIP 开关进行设置。在启动 CPU 期间，自动将参数分配给模块。这表示，例如，可以不分配新参数就交换模块。
- 此外，在硬件配置工具中可将参数分配给功能模块(FM)和通讯处理器(CP)，其分配方式与其它模块完全相同。每个 FM 和 CP (包含在 FM/CP 功能包中)都有与模块有关的对话框和规则。系统在对话框中只提供有效选项，以防止错误输入。

NetPro (网络配置)

可以使用 NetPro 通过 MPI 进行时间驱动的循环数据传送，操作如下：

- 选择通讯节点
- 在表中输入数据源和数据目标；自动产生要下载的所有块(SDB)，并自动完全下载到所有 CPU 中

也可以执行事件驱动的数据传送，操作如下：

- 设置通讯连接
- 从集成的块库中选择通讯或功能块
- 以选定的编程语言将参数分配给选中的通讯或功能块

1.3 STEP 7，5.3版本有何新特性？

下列主题区已经作了更新：

安装

- STEP 7 是面向 MS Windows 2000 Professional 和 MS Windows XP Professional 而发布的。
- 至于 STEP 7 V5.3，则存在一个新的授权过程。用户权限将不再通过授权来提供，而是通过许可证密钥来提供。许可证密钥将在自动许可证管理器中进行管理(参见通过自动许可证管理器授权的用户权限)。不再使用“AuthorsW”程序。

打印

所有应用程序中的页面尺寸和页面布局(页眉和页脚)现在均可使用**文件**(或特定应用程序中的相应菜单项) **> 页面设置**进行指定。这将不再一定要在SIMATIC管理器中集中完成。

SIMATIC管理器

- “比较块”对话框现在具有选项“比较详细资料”。要为进行比较的块选择路径，可点击“选择”按钮。
- 在“MMC 上的文件”文件夹中显示已保存到 MMC 或存储卡的数据，该文件夹位于块文件夹下。
- 对于库，增加了新的符号。所有由用户创建的 F 库，原来只能运行在 F-系统上，现在均支持。为创建 F 库，可选择**文件 > 新建 > 库**菜单命令。在显示的“新项目”对话框中，选择“F 库”复选框。
- 菜单命令 **PLC > 诊断/设置 > 节点闪烁测试**将使您能够通过闪烁的 FORCE LED 灯来识别直接连接到编程设备(PG)/PC 的节点(参见通过“可访问节点”窗口建立在线连接和诊断以太网(PROFInet)接口)。

对LAD/STL/FBD块编程

- “块的调用环境”对话框现在将允许您通过引用数据显示中的交叉引用功能来手动输入先前没有检测到的调用路径。
- 如果程序编辑器显示了当前的块状态，则您现在可按十进制、十六进制或浮点格式显示 LAD 和 FBD 语言下的当前状态值。
- 在选择**选项 > 自定义**之后，在显示的对话框中转到“常规”标签，指定每当达到可监控的块的最大数目时，是否应自动改变块的程序状态(参见通过“可访问节点”窗口建立在线连接)。

符号表

- 在符号表内部，您现在可选择和编辑连续区。这表示可复制和/或剪切部分符号表，并将其插入到另一个符号表中，或按要求将其删除。

组态和诊断硬件

- 先前的“H”可选软件包“S7-400H容错系统”将不再作为一个单独的可选软件包提供；取而代之，现在已将其集成在STEP 7 V5.3中。要打开相关的电子手册“S7-400H容错系统”，可在任务栏中选择**启动 > SIMATIC > 文档**。块库“冗余IO”包含了用于支持冗余I/O设备的块。
- 您现在可在硬件目录中搜索所期望的组件或任意的文本字符串(参见搜索硬件目录)。
- 现在可直接通过 Internet 地址搜索关于模块和组件的信息(产品支持、FAQ)。假如有可供使用的关于模块的信息，您将被带到包含有信息选择的页面。为此，可从硬件目录中或在模块机架上选择所需要的模块，右击打开关联菜单，上面列出了可用的信息选项(参见显示硬件目录中的组件信息)。
- 现在已支持具有新功能的模块：
  您可将参数分配给ET 200S的“选项处理”新功能(参见带选项处理的ET 200S)
  。
  您可给出CPU 41x-xxx40的位置标识号。

组态网络和连接

- NCM S7 工业以太网和 NCM S7 PROFIBUS，作为 S7 CP 的组态工具，现在已不再是单独的软件包，而是随 STEP 7 V5.3 一起自动安装。
- 网络视图现在可以切换为具有较小子网长度的视图。这样将提供一个更清晰的项目概况，特别是那些带有许多站的视图(参见创建和分配网络连接的参数和关于编辑网络组态的提示)。
- CPU 317-2 PN/DP 可通过集成的 PROFInet 接口作为客户机进行 S7 通讯。您可在标准库中查找到通讯块(通讯块、CPU 300，参见不同连接类型的块)。
- 错误和警告现在均显示在一个新设计的输出窗口中，用于进行一致性测试。该窗口已按列进行了显著的结构化处理，并具有用于对错误位置进行定位、显示与特定消息有关的帮助以及用于打印的菜单命令。
- 连接(包括跨越项目间连接)均可从具有功能“编译和下载对象”的中央点下载到站。如果用作起始点的对象是一个多项目，那么，连接涉及的所有站均将自动地下载，而不管站位于哪一个项目中(参见编译和下载对象)。

标准库

- 使用创建保留或非保留数据块的 SFC85 块以及用于 PROFInet 通讯的 SFC 112、SFC 113 和 SFC 114，已经对标准库“系统功能块”进行了扩展。
- 使用用于 CPU 317-2 PN/DP (CPU_300)的 S7 通讯的块，对标准库“通讯块”进行了扩展。
- 为库而使用的符号是新的。您现在可识别哪一个库是用户创建的 F 库以及哪些只能在 F 系统上运行。

过程诊断

- 新的**过程诊断 > 导入模板**菜单命令使您能够将用于过程监控的模板导入到S7-PDIAG中。

管理多语言文本

- 使用 STEP 7 V5.3，除了 CSV 格式外，您现在还可使用 XLS 格式作为导出格式。

1.4 STEP 7标准软件包的扩展使用

可以由软件选项包扩展标准软件包，选件包分成下列三类软件：

- 工程工具：这些工具为高级编程语言，以及技术含量较高的软件。
- 运行软件；这些软件包含现货供应软件，用于生产过程。
- 人机界面(HMI)；该软件专门用于操作员监控。

下表显示了不同可编程控制系统可使用的可选软件：

		STEP 7	
	S7-300 **S7-400**	**M7-300** **M7-400**	**C7-620**
工程工具			
• Borland C/C++		o	
• CFC	+1)	+	+2)
• DOCPRO	+	+3)	+
• HARDPRO	+		
• M7 ProC/C++		o	
• S7 GRAPH	+1)		+2)
• S7 HiGraph	+		+
• S7 PDIAG	+		
• S7 PLCSIM	+		+
• S7 SCL	+		+
• Teleservice	+	+	+
运行软件			
• 模糊控制	+		+

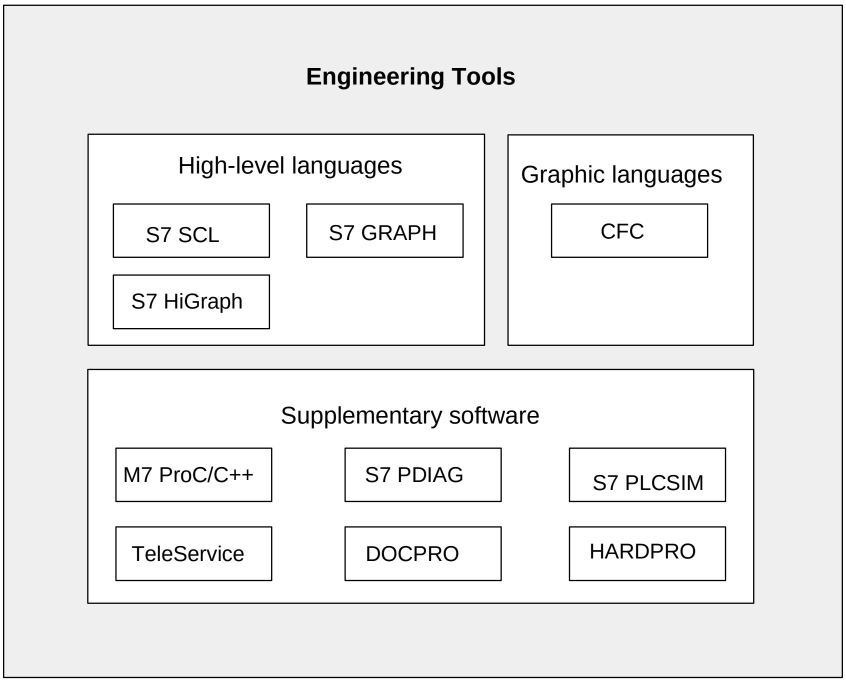
		STEP 7	
	S7-300 **S7-400**	**M7-300** **M7-400**	**C7-620**
• M7-DDE服务器		+	
• M7-SYS RT		o	
• 模块化PID控制	+		+
• PC-DDE服务器	+		
• PRODAVE MPI	+		
• 标准PID控制	+		+
人机界面			
• ProAgent			
• SIMATIC ProTool			
• SIMATIC ProTool/Lite			o
• SIMATIC WinCC			

o = 强制
+ =可选
[1] = 建议用于S7-400以上
[2] =不建议用于C7-620
[3] =不用于 C程序

1.4.1 工程工具

工程工具是面向任务的工具，可用来扩展标准软件包。工程工具包括：

- 程序员使用的高级语言
- 技术员工使用的图形语言
- 用于诊断、模拟、远程维护和设备文档等的辅助软件。



高级语言

下列语言在选件包中提供，可对 SIMATIC S7-300/S7-400 可编程逻辑控制器进行编程：

- S7 GRAPH 是用于对顺序控制(步和转移)进行编程的编程语言。在该语言中，过程顺序分成几个步。步包含控制输出的动作。由转移条件控制从一个步到另一个步的转移。
- S7 HiGraph 是一种编程语言，以状态图的形式描述异步、非顺序过程。为此，设备可分成几个独立功能单元，每个功能单元可处于不同状态。可通过在图形之间交换消息而使这些功能单元同步。
- S7 SCL 是符合 EN 61131-3 (IEC 1131-3)标准的基于文本的高级语言。它的语言结构与编程语言 C 和 Pascal 相似。因此，S7 SCL 尤其适用于熟悉高级语言编程的用户使用。比如，S7 SCL 可用于编程复杂或频繁发生的功能。

图形语言

用于 S7 和 M7 的 CFC 是以图形方式互连功能的编程语言。这些功能涉及范围非常大，从大量简单逻辑操作直至复杂控制和控制电路。在库中以块的形式提供大量该类功能块。通过将块复制到图表中，并用连接线将这些块互连，来进行编程。

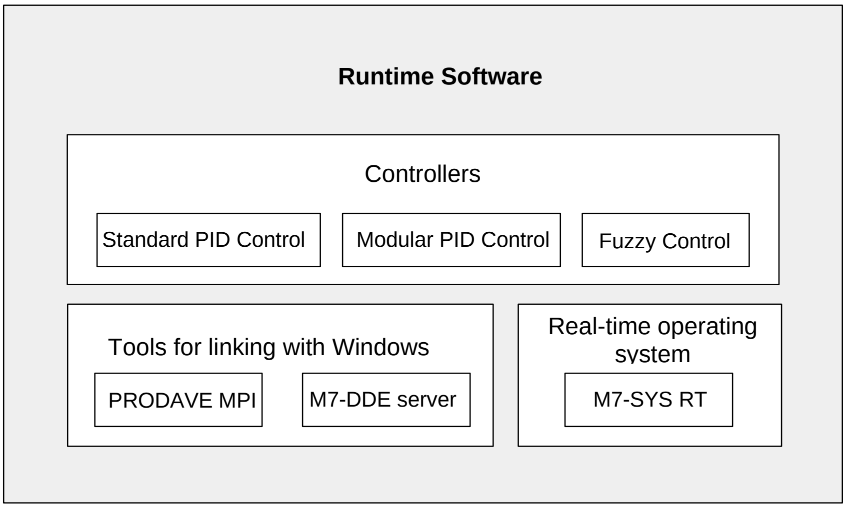
辅助软件

- Borland C++ (仅适用于 M7)包含 Borland 开发环境。
- 通过 DOCPRO，可以将 STEP 7 下创建的所有组态数据组织进接线手册。这些接线手册便于管理组态数据，并可根据特定要求准备打印信息。
- HARDPRO 是带用户支持的 S7-300 的硬件配置系统，用于组态大型复杂自动化任务。
- M7 ProC/C++ (仅适用于 M7)允许将编程语言 C 和 C++的 Borland 开发环境集成到 STEP 7 开发环境中。
- 可以使用 S7 PLCSIM (仅适用于 S7)模拟连接到编程设备或 PC 的 S7 可编程控制器，以进行测试。
- S7 PDIAG (仅适用于 S7)允许标准化组态 SIMATIC S7-300/S7-400 的过程诊断。过程诊断允许检测 PLC I/O 的故障和故障状态(例如，没有到达限位开关)。
- TeleService 是一种解决方案，可通过 PG/PC 的远程通讯网络，对远程 S7 和 M7 PLC 进行在线编程和维护。

1.4.2 运行软件

运行软件提供可在用户程序中调用的即时使用的解决方案，直接在自动化解决方案中执行。它包括：

- 用于 SIMATIC S7 的控制器，如标准、模块化和模糊逻辑控制
- 用于链接可编程控制器与 Windows 应用程序的工具
- 用于 SIMATIC M7 的实时操作系统



用于SIMATIC S7的控制器

- 标准 PID 控制允许将闭环控制器、脉冲控制器以及步骤控制器集成到用户程序中。带集成控制器设置的参数分配工具允许设置控制器，可在极短时间内优化使用。
- 如果简单 PID 控制器不足以解决自动化任务，请使用模块化 PID 控制。可以互连所包含的标准功能块，创建几乎任何一种控制器结构。
- 通过模糊控制，可以创建模糊逻辑系统。如果不能对过程进行数学定义或定义太过复杂，或者过程和顺控器没有按预期响应，或者发生线性化错误，或者同时又提供关于过程的信息，那么请使用这些系统。

用于链接Windows的工具

- PRODAVE MPI 是 SIMATIC S7、SIMATIC M7 和 SIMATIC C7 之间过程数据通讯量的工具栏。它自动控制通过 MPI 接口的数据流量。
- M7 DDE 服务器**(动态数据交换)**可用于将 Windows 应用程序链接到 SIMATIC M7 中的过程变量，而无需另外编程。

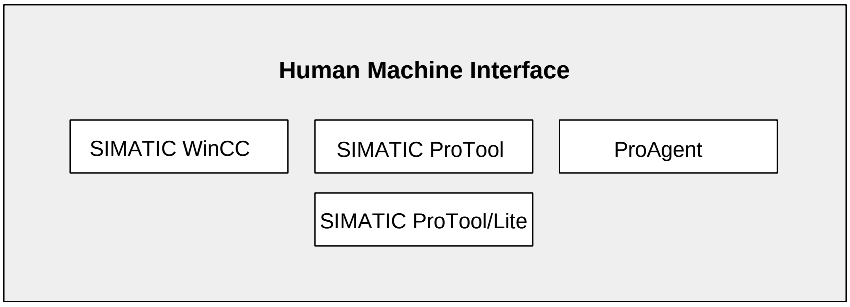
实时操作系统

- M7-SYS RT 包含操作系统 M7 RMOS 32 和系统程序。这是 SIMATIC M7 数据包使用 M7-ProC/C++和 CFC 的前提条件。

1.4.3 人机界面

人机界面(HMI)是专门设计用于在 SIMATIC 中进行操作员监控的软件。

- 开放式过程可视化系统 SIMATIC WinCC 是一个标准的操作员接口，包含所有可在任何工业领域、结合任何技术使用的重要的操作员监控功能。
- SIMATIC ProTool 和 SIMATIC ProTool/Lite 是用于组态 SIMATIC 操作面板(OP)和 SIMATIC C7 紧凑型设备的现代工具。
- ProAgent 是获取设备和机器中错误位置和原因信息的诊断软件，可提供快速、有针对的过程诊断。



2 安装

2.1 自动化许可证管理器

2.1.1 通过自动化许可证管理器获取用户权限

自动化许可证管理器

要使用 STEP 7 编程软件，需要一个产品专用的许可证密钥(用户权限)。从 STEP 7 V5.3 版本起，该密钥通过自动化许可证管理器安装。

自动化许可证管理器是 Siemens AG 的软件产品。它用于管理所有系统的许可证密钥(许可证模块)。

自动化许可证管理器位于下列位置：

- 在要求许可证密钥的软件产品的安装设备上
- 在单独的安装设备上
- 从 Internet 上 Siemens AG 的 A&D 客户支持页面下载

自动化许可证管理器集成了自身的在线帮助。要在安装许可证管理器后获取帮助，请按F1或选择**帮助 > 许可证管理器帮助**。该在线帮助包含自动化许可证管理器功能和操作的详细信息。

许可证

合法使用受许可证保护的 STEP 7 程序软件包时必须要有许可证。许可证为用户提供使用产品的合法权限。下列各项提供使用权限证明：

- **CoL (许可证证书)**，和
- 许可证密钥

许可证证书(CoL)

产品所包含的“许可证”是使用该产品权限的合法证明。该产品只能供许可证证书(CoL)拥有者或由拥有者授权使用的人员使用。

许可证密钥

许可证密钥是软件使用许可证的技术表示(电子“许可证标志”)。
SIEMENS AG给受许可证保护的所有软件颁发许可证密钥。启动计算机后，只能在确认具有有效许可证密钥之后，才能根据许可证和使用条款使用该软件。

注意

- 可以使用不带许可证密钥的标准软件来熟悉用户接口和功能。
- 但是，必须使用许可证才能根据许可证协议完全无限制地使用STEP 7 软件。
- 如果还**没有**安装许可证密钥，那么将定期提示您安装许可证密钥。

可以按如下所述，在各种类型的存储设备之间存储和传送许可证密钥：

- 在许可证密钥软盘上
- 在本地硬盘上
- 在网络硬盘上

如果安装没有提供许可证的软件产品，则请判定需要何种许可证密钥，并按要求订购。

欲知获取和使用许可证密钥的详情，请参见自动化许可证管理器的在线帮助。

许可证类型

给西门子AG软件产品提供下列不同类型的面向应用的用户许可证。软件的实际特性取决于所安装的许可证密钥类型。可在附带的许可证证书中获得使用类型。

许可证类型	描述
单独许可证	该软件可在希望具有无限使用时间的单台计算机上使用。
浮动许可证	该软件可在希望具有无限使用时间的计算机网络(“远程使用”)上使用。
试用许可证	该软件可在下列限制条件下使用： • 有效期最多为14天， • 第一次使用之日起的总操作天数， • 用于测试和确认(免除责任)。
升级许可证	在软件升级方面，现有系统中的特定要求可能适用： • 升级许可证可用于将“旧版本X”软件转换为新版本X+。 • 由于给定系统中需处理的数据量增大，可能需要升级。

2.1.2 安装自动化许可证管理器

自动化许可证管理器通过MSI设置过程安装。STEP 7产品CD包含自动化许可证管理器的安装软件。
可以在安装STEP 7的同时安装自动化许可证管理器或在以后安装。

注意

- 欲知如何安装自动化许可证管理器的详细信息，请参见当前的“Readme.wri”文件。
- 自动化许可证管理器的在线帮助包含许可证密钥功能和处理所需的所有信息。

随后安装许可证密钥

启动 STEP 7 软件时如果没有可用的许可证密钥，将显示一个指示该情况的警告消息。

注意

- 可以使用不带许可证密钥的标准软件来熟悉用户接口和功能。
- 但是，必须使用许可证才能根据许可证协议完全无限制地使用STEP 7软件。
- 如果还**没有**安装许可证密钥，那么将定期提示您安装许可证密钥。

可按下列方法随后安装许可证密钥：

- 从软盘上安装许可证密钥
- 安装从 Internet 上下载的许可证密钥。这种情况下，必须首先订购许可证密钥。。
- 使用网络中可用的浮动许可证密钥

欲知安装许可证密钥的详细信息，请参见自动化许可证管理器的在线帮助。请按F1或选择菜单命令**帮助 > 许可证管理器帮助**。

注意

- 在Windows 2000/XP中，只有在本地硬盘上安装并具有写访问状态时，才能操作许可证密钥授权。
- 也可以在网络内使用浮动许可证(“远程”使用)。

2.1.3 处理许可证密钥的指南

警告

请注意在自动化许可证管理器在线帮助以及在安装CD-ROM中STEP 7 Readme.wri文件中关于处理许可证密钥的信息。如果不遵守这些指南，那么将丢失许可证密钥且不可恢复。

要访问自动化许可证管理器的在线帮助，请按 F1 获取上下文关联帮助或选择**帮助 > 许可证管理器帮助**菜单命令。

该帮助部分包含许可证密钥功能和处理所需的所有信息。

2.2 安装STEP 7

STEP 7 安装程序可自动完成安装。通过菜单可控制整个安装过程。可通过标准 Windows 2000/XP/Server 2003 软件安装程序执行安装。

安装的主要步骤为：

- 将数据复制到编程设备中
- 组态 EPROM 和通讯驱动程序
- 输入标识号
- 安装许可证密钥(如果需要)

注意

西门子编程设备装运时在硬盘上包含可即时安装的STEP 7软件。

安装要求

- 操作系统：
  Microsoft Windows 2000 或 Windows XP。
- 基本硬件：
  包含下列各项的编程设备或PC：
- 奔腾处理器(对于Windows 2000，使用P233；对于 Windows XP，使用P233)
- 至少128MB RAM。
- 彩色监视器，键盘和鼠标、Microsoft Windows支持所有这些组件

编程设备(PG)是具有特殊紧凑型设计、用于工业用途的 PC。它配备齐全，可用来对 SIMATIC PLC 进行编程。

- 硬盘空间：
  请参见“README.WRI”文件，获取所需硬盘空间信息。
- MPI 接口(可选)：
  只有在STEP 7下通过MPI与PLC通讯时才要求使用MPI接口来互连PG/PC和PLC。

  此时需要：
- 一个PC适配器以及一根与设备通讯端口相连的假调制解调器(RS232)，或
- 在设备中安装MPI模块(例如，CP5611)。

PG装配有**MPI**接口。

- 外部存储器(可选)
  只有在通过PC编程EPROM时，才要求使用外部存储器。

注意

请参见README.WRI文件和“与标准STEP 7软件包版本兼容的SIMATIC软件包列表”，获取关于STEP 7安装的信息。

可以在开始菜单**开始 > Simatic > 产品说明**下找到自述文件。

兼容性列表可以通过“开始”菜单，在**开始 > Simatic > 文档**下找到。

2.2.1 安装过程

准备安装

开始安装软件之前，必须启动操作系统(Windows 2000/XP)。

- 如果已经在 PG 的硬盘上保存有可安装的 STEP 7 软件，那么不需要外部存储介质。
- 若要从 CD-ROM 中安装，请在 PC 的 CD-ROM 驱动器中插入 CD-ROM。

启动安装程序

按如下所述操作，安装软件：

1. 插入CD-ROM，双击“SETUP.EXE”文件。
2. 按照屏幕上安装程序的逐步指示进行安装。

该程序引导您完成安装的所有步骤。可以前进到下一步或返回上一步。

安装期间，对话框提示从显示的选项中进行选择。下列注意事项有助于快速、方便地正确选择安装。

如果已经安装某一种版本的STEP 7...

如果安装程序在编程设备上检测到其它版本的 STEP 7，则会显示相应消息。然后可以选择：

- 中止安装(从而可以在 Windows 下卸载旧 STEP 7 版本然后重新启动安装，或，
- 继续执行安装，覆盖以前版本。

为进行良好的软件管理，始终应该在安装新版本之前卸载任何旧版本。

用新版本覆盖旧版本的缺点是随后卸载旧软件版本时，旧版本的一些组件可能不能删除。

选择安装选项提供三个选项，选择安装范围：

- 标准安装：用于用户界面的所有对话框语言、所有应用以及所有实例。请参见当前产品信息，获取该类型组态所要求的存储器空间信息。
- 基本安装：只有一种对话框语言，没有实例。请参见当前产品信息，获取该类型组态所要求的存储器空间信息。
- 用户自定义(“自定义”) 安装您可以确定安装范围，例如，程序、数据库、实例和通讯功能。

ID号

在安装期间，您会被提示输入一个 ID 号(可在软件产品证书或许可证密钥软盘中获得)。

安装许可证密钥

安装期间，程序检查是否在硬盘上安装了相应的许可证密钥。如果没有找到有效的许可证密钥，将会显示一条消息，指示必须具有许可证密钥才能使用该软件。根据需要，可以立即安装许可证密钥或者继续执行安装、以后再安装许可证密钥。如果希望马上安装许可证密钥，那么在提示插入授权软盘时，请插入授权软盘。

PG/PC接口设置

安装期间，会显示一个对话框，在此可以将参数分配给编程设备/PC 接口。更多信息，请参见“设置 PG/PC 接口”。

将参数分配给存储卡

安装期间，会显示一个对话框，可以将参数分配给存储卡。

- 如果不使用存储卡，则不需要 EPROM 驱动程序。选择“无 EPROM 驱动程序”选项。
- 否则，选择适用于 PG 的条目。
- 如果使用 PC，请选择外部编程器的驱动程序。在此，必须指定连接该编程器的端口(例如，LPT1)。

通过在 STEP 7 程序组或控制面板中调用“存储卡参数分配”程序，可以在安装后修改设定的参数。

闪存文件系统

在分配存储卡参数的对话框中，可以选择安装闪存文件系统。

例如，在 SIMATIC M7 下，当将单个文件写入到 EPROM 存储卡，而不修改存储卡的其它内容时，要求使用闪存文件系统。

如果使用合适的编程设备(PG 720/PG 740/PG 760、现场 PG 和专业 PG)或外部编程器，并希望使用闪存功能，那么请安装闪存文件系统。

如果在安装期间发生错误

可能由于下列原因取消安装：

- 如果在启动安装之后立即发生初始化错误，那么极有可能没有在 Windows 下启动*安装*。
- 硬盘空间不足：不管安装范围如何，标准软件安装的最小自由硬盘空间要求为 100MB。
- 故障 CD-ROM：如果 CD 发生故障，请与当地的西门子代表处联系。
- 操作员错误。仔细按照指示，重新启动安装。

已经完成安装后…

屏幕消息报告安装成功。

如果在安装期间修改了系统文件，则会提示重启 Windows。重启 (热启动) 后，可以启动 STEP 7 应用程序、SIMATIC 管理器。

成功安装后，就会建立 STEP 7 程序组。

2.2.2 设置PG/PC接口

在此，可以组态PG/PC和PLC之间的通讯。安装期间，将显示一个对话框，可以将参数分配给PG/PC接口。也可以在STEP 7程序组中调用“设置PG/PC接口”，在安装后打开该对话框。这样可以在安装以后修改接口参数，而与安装无关。

基本步骤

要操作接口，必须执行下列各项：

- 在操作系统中组态
- 合适的接口组态

如果使用带 MPI 卡或通讯处理器(CP)的 PC，那么应该在 Windows 的“控制面板”中检查中断和地址分配，确保没有发生中断冲突，也没有地址区重叠现象。

在 Windows 2000 和 Windows XP 中，不再支持 ISA 组件 MPI-ISA 卡，因此安装时不再提供该组件。

为简化将参数分配给编程设备/PC 接口，对话框将显示缺省的基本参数设置(接口组态)选择列表。

将参数分配给PG/PC接口

步骤(可在在线帮助中获得详细信息)：

1. 在Windows“控制面板”中双击“设置PG/PC接口”。
2. 将“应用访问点”设置为“S7ONLINE”。
3. 在“使用的接口参数设置”列表中，选择所要求的接口参数设置。如果没有显示所要求的接口参数设置，那么必须首先通过“选择”按钮安装一个模块或协议。然后自动产生接口参数设置。在即插即用系统中，不能手动安装即插即用CP (CP 5611和 CP 5511)。在PG/PC中安装硬件后，它们自动集成在“设置PG/PC接口”中。
   - 如果选择具有**自动识别总线参数**功能的接口(例如CP 5611 (自动))，那么可以将编程设备或PC连接到MPI或PROFIBUS，而无需设置总线参数。如果传输率 < 187.5 Kbps，那么读取总线参数时，可能产生高达1分钟的延迟。
     自动识别的要求：将循环广播总线参数的主站连接到总线。所有新MPI组件都如此操作；对于PROFIBUS子网，必须启用循环广播总线参数(缺省的PROFIBUS网络设置)。

4. 如果选择了一个**不能自动识别总线参数**的接口，那么可以显示其属性，然后进行修改，使其与子网相匹配。

如果与其它设置发生冲突(例如，中断或地址分配)，那么也必须进行修改。此时，可在 Windows 的硬件识别和控制面板中作一些相应修改(参见下面)。

警告

请勿从接口设置中删除任何“TCP/IP”参数。

否则将引起其它应用故障。

检查中断和地址分配

如果使用带 MPI 卡的 PC，则应该始终检查缺省中断和缺省地址区是否为空闲，如有必要，选择一个空闲的中断和/或地址区。

Windows 2000

在 Windows 2000 下，可以：

- 在**控制面板 > 管理工具 > 计算机管理 > 系统工具 > 系统信息 > 硬件资源**下查看资源。
- 在**控制面板 > 管理工具 > 计算机管理 > 系统工具 > 设备管理器 > SIMATIC NET > CP 名称 > 属性 > 资源**下改变资源

Windows XP

在 Windows XP 下，可以

- 在**开始 > 所有程序 > 附件 > 系统 > 系统程序 > 系统信息 > 硬件资源**下查看资源。
- 在**控制面板 > 桌面 > 属性 > 设备管理器 > SIMATIC NET > CP 名称 > 属性 > 资源**下改变资源。

2.3 卸载STEP 7

2.3.1 卸载STEP 7

使用标准 Windows 方法，卸载 STEP 7：

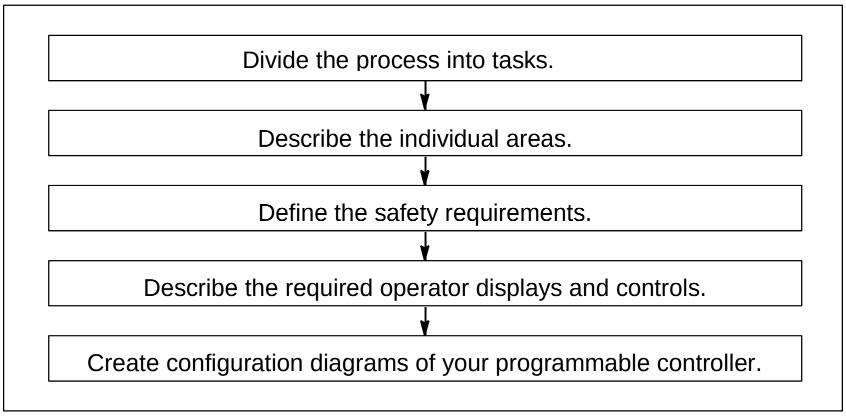
1. 在“控制面板”中双击“添加/删除程序”图标，启动Windows软件安装对话框。
2. 在已安装软件的显示列表中选择STEP 7条目。点击“添加/删除”按钮。
3. 出现“删除共享文件”对话框时，如果不确定，则请点击“否”按钮。

3 详述自动化概念

3.1 规划自动化项目的基本过程

该章概述了规划可编程控制器(PLC)自动化项目所涉及的基本任务。基于自动化控制一个工业混料过程的实例，逐步引导您完成整个过程。

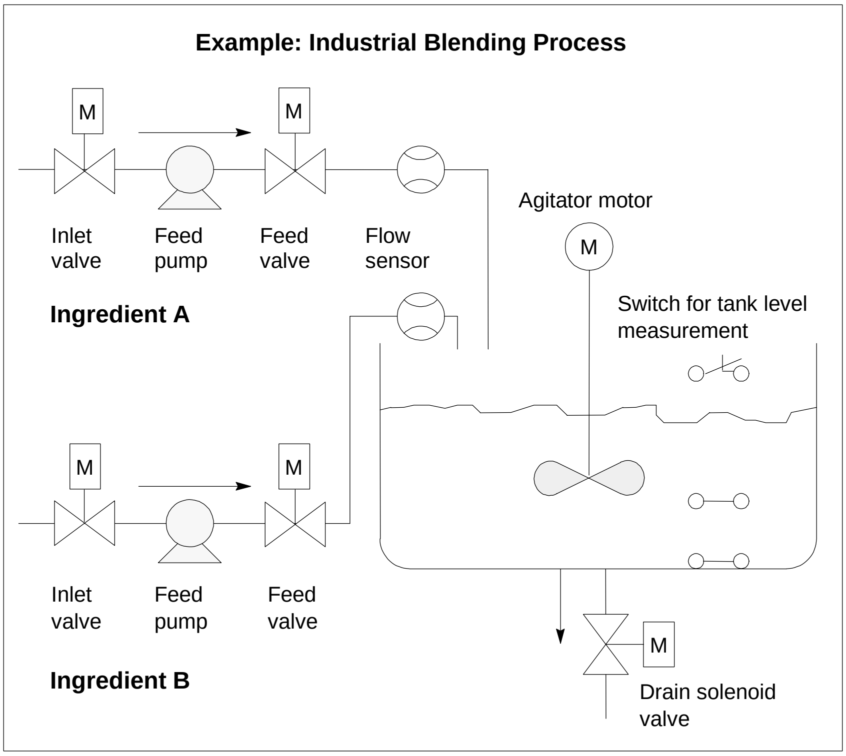
规划自动化项目有多种方法。下图阐述了可用于任何项目的基本步骤。



3.2 将过程分成任务和区域

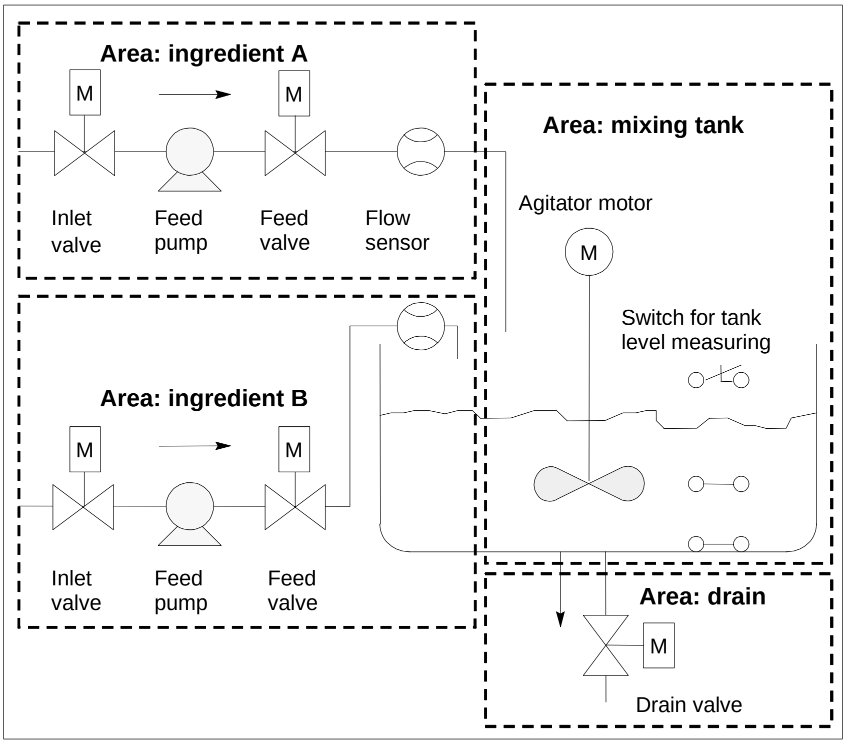
自动化过程包含大量单个任务。通过在过程内识别相关任务组，然后将这些组分成更小的任务，用这种方法甚至可以定义最为复杂的过程。

下面的工业混料过程实例可用于阐述如何将过程划分为一些功能区域和单个任务：



确定过程区域

定义要控制的过程后，将项目分成相关的组或区域：



由于每组都分成小型任务，因此要求控制部分过程的任务变得不太复杂。

在工业混料过程实例中，可以识别 4 个不同区域(参见下表)。在该实例中，配料 A 区域包含的设备与配料 B 区域相同。

功能区域	所使用的设备
配料A	配料A的进料泵 配料 A的入口阀 配料A的进料阀 配料 A的流量传感器
配料B	配料B的进料泵 配料 B的入口阀 配料B的进料阀 配料 B的流量传感器
混料罐	搅拌器电机 罐液位测量开关
排料	排料阀

3.3 描述单个功能区

在过程内描述每个区域和任务时，不仅需要定义每个区域的操作，还需要定义控制该区域的不同元件这些元件包括：

- 每个任务的电气、机械和逻辑输入和输出
- 单个任务之间的互锁和依赖性

工业混料过程实例使用泵、电机和阀。必须精确描述识别操作期间所要求的操作特性和互锁类型。下表提供了描述工业混料过程中所使用设备的实例。完成描述后，也可以使用它来订购需要的设备。

配料A/B：进料泵电机
进料泵电机将配料A和B传送到混料罐。 • 流速：每分钟400l (100加仑) • 额定值：1200rpm时为100kW (134hp)
通过混料罐附近的操作员站控制泵(启动/停止)。计数启动次数以用于维护。可通过一个按钮将计数器和显示器复位。
要操作泵，必须满足下列条件： • 混料罐不满。 • 闭合混料罐的排料阀。 • 没有激活紧急断电。
如果满足下列条件，泵将关闭： • 启动泵电机7秒后，流量传感器指示无流量。 • 流量传感器指示停止流动。

配料A/B：入口和进料阀
1. 配料A和B的入口阀和进料阀可允许或防止配料流入混料罐中。阀有一个具有弹簧复位的螺线管。 • 激活螺线管时，打开阀。 • 取消激活螺线管时，闭合阀。
由用户程序控制入口阀和进料阀。
要激活阀，必须满足下列条件： • 进料泵电机已经运行1秒以上。
如果满足下列条件，泵将关闭： • 流量传感器指示无流量。

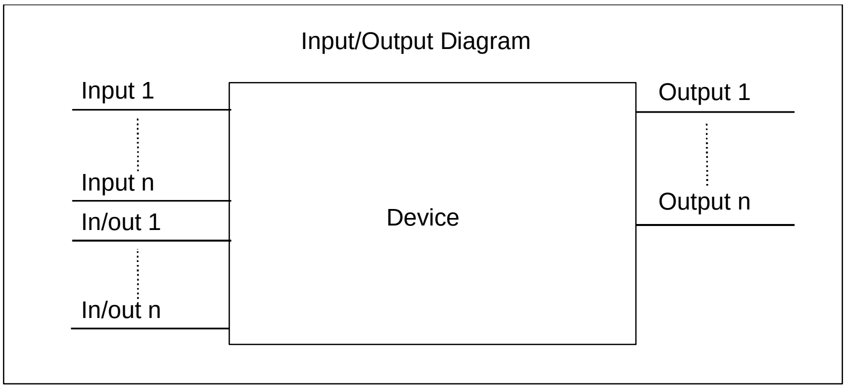
搅拌器电机
2. 搅拌器电机在混料罐中混合配料A和配料B。 • 额定值：1200rpm时为100kW (134hp)
通过混料罐附近的操作员站控制搅拌器电机(启动/停止)。计数启动次数以用于维护。可通过一个按钮将计数器和显示器复位。
要操作泵，必须满足下列条件： • 罐液位传感器没有指示“罐液位低于最小值”。 • 混料罐的排料阀闭合。 • 没有激活紧急断电。
如果满足下列条件，泵将关闭： • 流速计在启动电机后10秒内不指示已经到达额定速度。

排料阀
排料阀允许将混料物(通常为重力进料)排放到过程中的下一个阶段。阀有一个具有弹簧复位的螺线管。 • 激活阀时，打开出口阀。 • 取消激活螺旋管时，闭合出口阀。
通过操作员站控制出口阀(打开/关闭)。
3. 可在下列条件下打开排料阀： • 搅拌器电机关闭。 • 罐液位传感器没有指示“罐空” • 没有激活紧急断电。
如果满足下列条件，泵将关闭： • 罐液位传感器指示“罐空”。

罐液位测量开关
1. 混料罐中的开关指示罐中的液位，用于互锁进料泵和搅拌机电机。

3.4 列出输入、输出和输入/输出

写完要控制的每个设备的物理描述后，需绘制每个设备或任务区域的输入和输出图
。



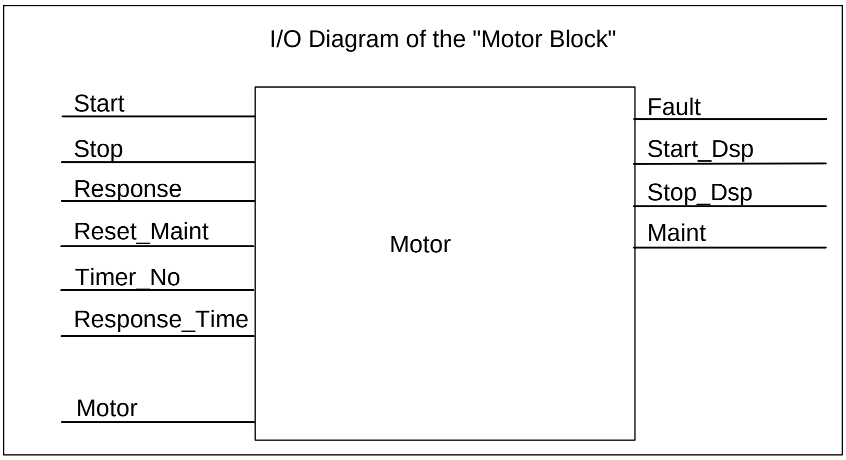
这些图与要编程的逻辑块相一致。

3.5 创建电机的I/O图

在工业混料过程的实例中使用两个进料泵和一个搅拌机。每个电机都由其自身的“电机块”控制，该块对于所有三个设备都相同。该块要求 6 个输入：两个输入用于启动或停止电机，一个输入用于复位维护显示器，一个输入用于电机响应信号(电机运行/不运行)，一个输入用于时间，在该时间内必须接收响应信号，一个输入用于测量时间的计时器数目。

逻辑块还要求四个输出：两个输出指示电机的操作状态，一个输出指示故障，另一个输出用于指示应该开始维护电机。

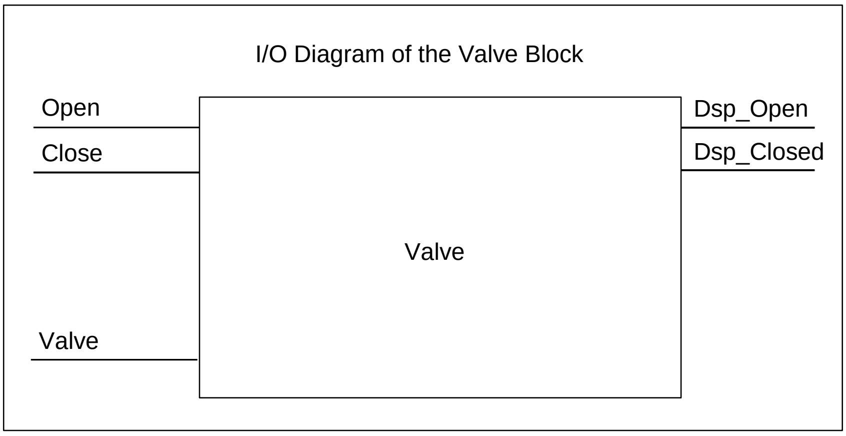
需要输入/ 输出来激活电机。它用于控制电机，但同时也可在“电机块”的程序中进行编辑和修改。



3.6 创建阀的I/O图

每个阀都由其自身的“阀块”控制，该块对于所使用的所有阀都相同。逻辑块有两个输入：一个输入用于打开阀，一个输入用于关闭阀。它还有两个输出：一个输出用于指示阀打开，另一个输出用于指示阀闭合。

阀块有一个输入/输出，用于激活阀。它用于控制阀，但同时也可在“阀块”的程序中进行编辑和修改。



3.7 建立安全要求

根据法律要求和人身健康及安全政策，确定需要哪些附加元件，以确保过程安全。在描述中，还应该包括安全元件对过程区域的所有影响。

定义安全要求

查找要求硬件电路满足安全要求的设备类型。通过定义，这些安全电路可独立于可编程控制器进行操作(虽然安全电路通常提供I/O接口，允许与用户程序协调操作)。通常，可以组态矩阵式，在其自身的紧急断电范围内连接每个执行器。该矩阵式是安全电路的电路图基础。

按如下执行来设计安全机制：

- 确定单个自动化任务之间的逻辑和机械/电气互锁。
- 设计电路，允许在紧急情况下手动操作属于该过程的设备。
- 为确保操作过程安全，需建立更多安全要求。

创建安全电路

工业混料过程实例使用下列逻辑电路作为安全电路：

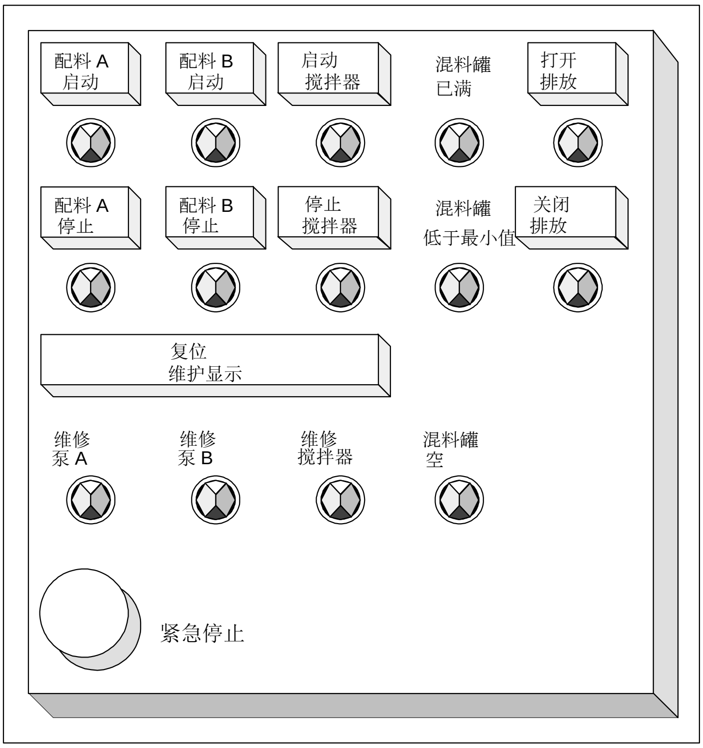
- 紧急断电开关切断下列设备电源，与可编程控制器(PLC)无关：
  - 配料 A 的进料泵
  - 配料 B 的进料泵
  - 搅拌器电机
  - 阀
- 紧急断电开关位于操作员站上。
- 控制器的一个输入指示紧急断电开关的状态。

3.8 描述所需要的操作员显示和控件

每个过程都要求有一个操作员界面，允许人员进行干预。部分设计规范包括操作员控制台设计。

定义操作员控制台

在实例所述的工业混料过程中，可由操作员控制台上的按钮启动或停止每个设备。该操作员控制台包括显示操作状态的指示灯(参见下表)。



控制台还包括显示灯，用于指示经过一定启动次数后要求维护的设备，以及紧急断电开关，通过该开关可立即终止过程。控制台还有一个复位按钮，用于三台电机的维护显示。通过该按钮，可以关闭指示应该维护电机的维护显示灯，并将相应的计数器复位到 0。

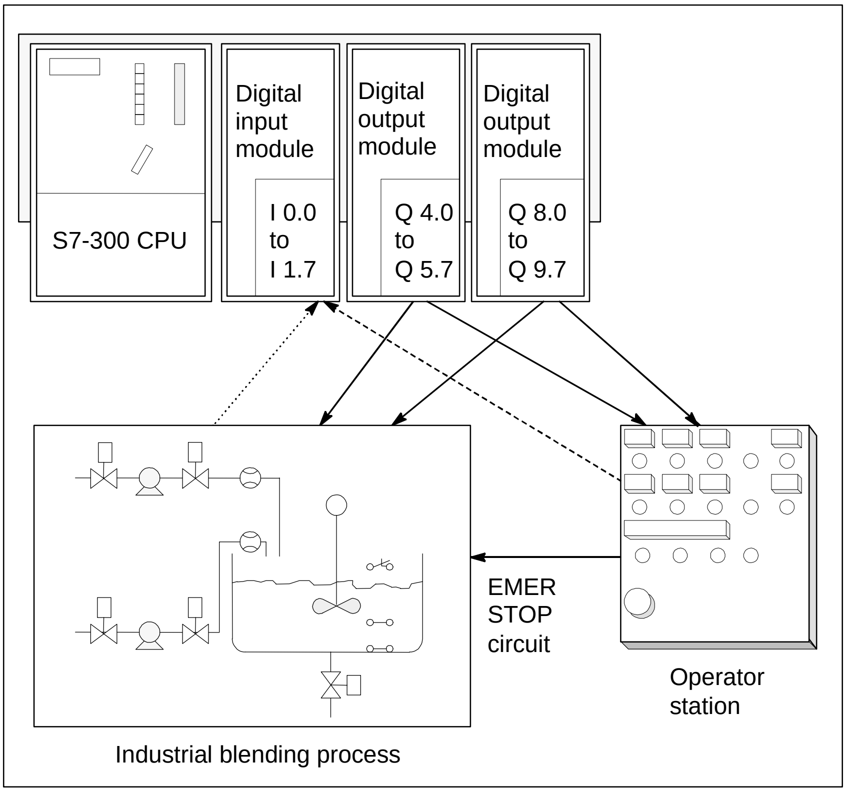
3.9 创建组态图

将设计要求文档化后，必须决定项目所要求的控制设备类型。

通过确定希望使用哪些模块，还可以确定可编程控制器的结构。创建一个确定下列各项的组态图：

- CPU 型号
- I/O 模块的编号和类型
- 组态物理输入和输出

下图阐述了用于工业混料过程的 S7 组态实例。



4 设计程序结构的基本原理

4.1 CPU中的程序

CPU 原则上运行两个不同的程序：

- 操作系统
- 用户程序。

操作系统

每个 CPU 都带有集成的操作系统，组织与特定控制任务无关的所有 CPU 功能和顺序。操作系统任务包括下列各项：

- 处理重启(热启动)和热重启。
- 更新输入的过程映像表，并输出输出过程映像表
- 调用用户程序
- 采集中断信息，调用中断 OB。
- 识别错误并进行错误处理
- 管理存储区域
- 与编程设备和其它通讯伙伴进行通讯

通过修改操作系统参数(操作系统缺省设置)，可以在某些区域影响 CPU 响应。

用户程序

可以创建用户程序，并将其下载到 CPU 中。它包含处理特定自动化任务所要求的所有功能。用户程序任务包括：

- 确定 CPU 的重启(热启动)和热重启条件(例如，用特定值初始化信号)
- 处理过程数据(例如，产生二进制信号的逻辑链接，获取并评估模拟量信号，指定用于输出的二进制信号，输出模拟值)
- 响应中断
- 处理正常程序周期中的干扰。

4.2 用户程序中的块

4.2.1 用户程序中的块

STEP 7 编程软件允许您构造用户程序，即，将程序分成单个、独立的程序段。这具有下列优点：

- 大程序更易于理解。
- 可以标准化单个程序段。
- 简化程序组织。
- 更易于修改程序。
- 可测试单个程序段，因而简化调试。
- 系统调试变得更简单。

工业混合过程实例阐述了将一个自动化过程分成单个任务的优点。结构化用户程序的程序段，即程序块对应于这些单个任务。

块类型

在 S7 用户程序内可使用多种类型的块：

块	功能简介	参见
组织块(OB)	OB确定用户程序的结构。	组织块和程序结构
系统功能块(SFB)和系统功能(SFC)	SFB和SFC集成在S7 CPU中，使您可以访问一些重要的系统功能。	系统功能块(SFB)和系统功能(SFC)
功能块(FB)	FB是带有用户可自行编程的“存储器”的块。	功能块(FB)
功能(FC)	FC包含频繁使用的功能的例行程序。	功能(FC)
背景数据块(instance DB)	调用FB/SFB时，背景数据块与块关联。它们在编译期间自动创建。	背景数据块
数据块(DB)	DB是用于存储用户数据的数据区。除分配给功能块的数据外，共享数据块也可由任何一个块来定义和使用。	共享数据块(DB)

OB、FB、SFB、FC和SFC包含程序段，因此也称为逻辑块。每种块类型许可的块的数目和块的长度由CPU决定。

4.2.2 组织块和程序结构

组织块(OB)表示操作系统和用户程序之间的接口。组织块由操作系统调用，控制循环中断驱动的程序执行、PLC启动特性和错误处理。可以对组织块进行编程来确定CPU特性。

组织块优先级

组织块确定单个程序段执行的顺序(启动事件)。一个OB调用可以中断另一个OB的执行。哪个OB允许中断另一个OB取决于其优先级。高优先级的OB可以中断低优先级的OB。背景OB的优先级最低。

中断类型和优先级

启动事件触发OB调用称为中断。下表显示了STEP 7 中的中断类型以及分配给这些中断的组织块的优先级。不是所有的S7 CPU都提供下表所列的所有组织块以及优先级(参见“S7-300可编程控制器，硬件和安装手册”以及“S7-400、M7-400可编程控制器模块规范参考手册”)。

中断类型	组织块	优先级(缺省)	参见：
主程序扫描	OB1	1	用于循环程序处理的组织块(OB1)
日历中断	OB10至OB17	2	日历中断组织块(OB10至OB17)
延时中断	OB20 OB21 OB22 OB23	3 4 5 6	延时中断组织块(OB20至OB23)
循环中断	OB30 OB31 OB32 OB33 OB34 OB35 OB36 OB37 OB38	7 8 9 10 11 12 13 14 15	循环中断组织块(OB30至OB38)

中断类型	组织块	优先级(缺省)	参见：
硬件中断	OB40 OB41 OB42 OB43 OB44 OB45 OB46 OB47	16 17 18 19 20 21 22 23	硬件中断组织块(OB40至OB47)
DPV1 中断	OB 55 OB 56 OB 57	2 2 2	编程DPV1设备
多计算中断	OB60 多值计算	25	多值计算 - 多个CPU的同步操作
同步循环中断	OB 61 OB 62 OB 63 OB 64	25	在PROFIBUS-DP上组态短的等长度过程响应时间
冗余错误	OB70 I/O 冗余错误(仅在H系统中) OB72 CPU 冗余错误(仅在H系统中)	25 28	“错误处理组织块”(OB70至OB87 / OB121至OB122)
异步错误	OB80 时间错误 OB81电源错误 OB82诊断错误 OB83插入/删除模块中断 OB84 CPU 硬件故障 OB 85 程序周期错误 OB86 机架故障 OB87 通讯错误	25 (如果在启动程序中出现异步错误OB，那么为28)	错误处理组织块(OB70至OB87 / OB121至OB122)
背景循环	OB90	29 1)	背景组织块(OB90)
启动	OB100重启动(热重启动) OB101热重启动 OB102冷重启动	27 27 27	启动组织块(OB100/OB101/OB102)
同步错误	OB121编程错误 OB122访问错误	引起错误的OB的优先级	错误处理组织块(OB70至OB87 / OB121至OB122)
1) 优先级29与优先级0.29一致。背景周期比自由周期的优先级低。			

修改优先级

可以通过STEP 7给中断分配参数。例如，通过参数分配，可以取消选定参数块中的中断OB或优先级：日历中断、延时中断、循环中断和硬件中断。

S7-300 CPU 上的组织块优先级固定。

对于 S7-400 CPU (和 CPU 318)，可以通过 STEP 7 修改下列组织块的优先级：

- OB10 至 OB47
- 运行模式中的 OB70 至 OB72 (仅适用于 H CPU) 和 OB81 至 OB87。

允许下列优先级：

- 优先级 2- 23，用于 OB10 至 OB47
- 优先级 2 - 28，用于 OB70 至 OB72
- 优先级 24 - 26，用于 OB81 至 OB87；对于大约在 2001 年的年中所生产的 CPU (固件版本 V3.0)，其优先级范围可以扩展：优先级 2 – 26 可用于 B 81 - OB 84 以及 OB 86 和 OB 87。

可以将相同优先级分配给多个 OB。具有相同优先级的 OB 按照其启动事件发生的先后次序进行处理。

由同步错误启动的错误 OB，其执行优先级与块发生错误时的执行优先级相同。

本地数据

创建逻辑块(OB、FC、 FB)时，可以声明临时本地数据。CPU 上的本地数据区可以分成不同优先级。

在 S7-400 上，可以通过 STEP 7 在“优先级”参数块中修改每个优先级的本地数据量。

OB的启动信息

每个组织块都有20个字节的本地数据的启动信息，在启动OB时，由操作系统提供。启动信息指定OB的启动事件、OB启动的日期和时间、所发生的错误以及诊断事件。

例如，OB40 是硬件中断 OB，其启动信息包含产生中断的模块地址。

取消选定中断OB

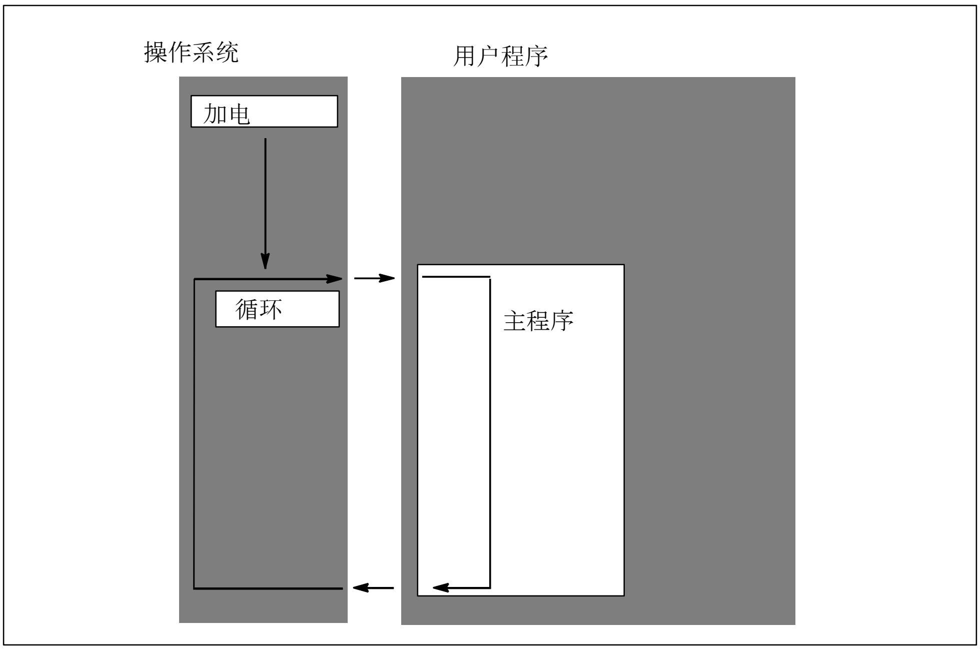
如果将优先级为0 或少于20个字节的本地数据分配给优先级，则将取消选定相应的中断OB。处理取消中断OB的限制条件如下：

- 处于运行模式时，这些中断 OB 不能复制或链接到用户程序。
- 处于停止模式时，可以将它们复制或链接到用户程序，但 CPU 执行重启(热启动)时，它们停止启动，并向诊断缓冲区输入一个条目。

通过取消选定不需要的中断 OB，可以增大可用的本地数据区容量，从而可以保存其它优先级的临时数据。

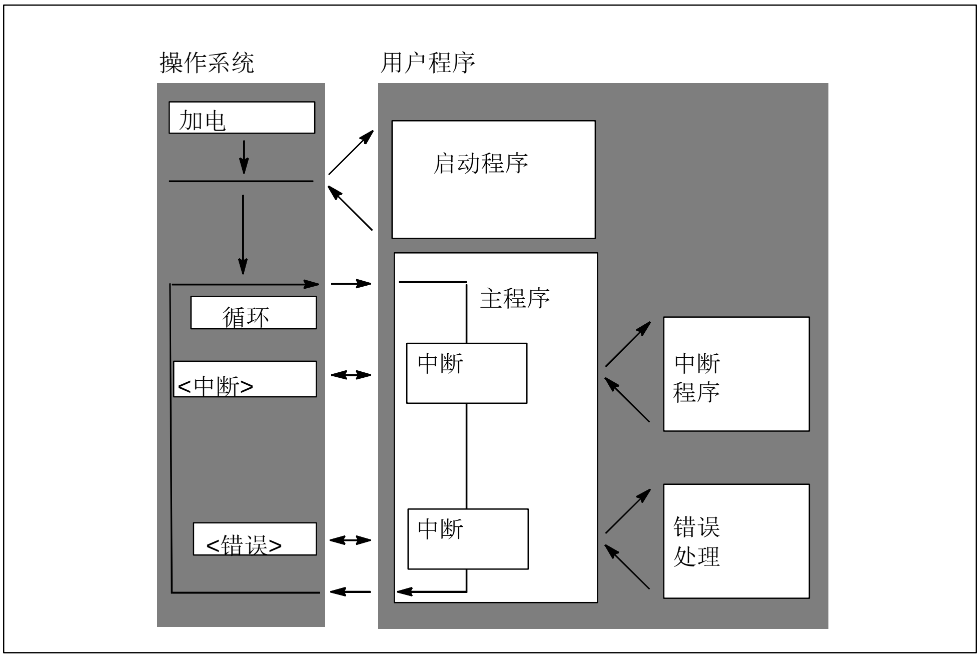
循环程序处理

循环程序处理是可编程逻辑控制器上“正常”执行的程序类型，表示操作系统在程序循环(周期)中运行，在每次循环中，都会调用主程序中的组织块OB1。即循环执行OB1中的用户程序。



事件驱动的程序处理

可由特定的事件(中断)中断循环程序处理。如果发生该类事件，将在命令边界中断当前执行的块，然后调用分配给该特定事件的其它组织块。该组织块一旦执行，将在中断点继续执行循环程序。

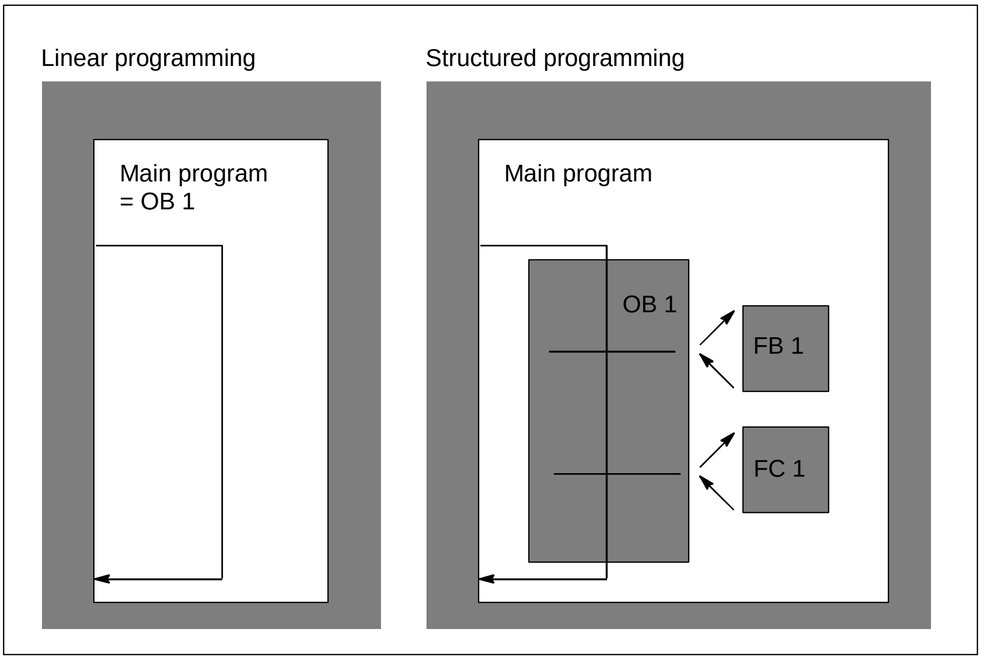


这表示可以处理部分用户程序，这些用户程序只有在需要时才循环处理。用户程序可分成一些“子程序”，分布在不同的组织块中。如果用户程序要对相对较少发生的重要信号(例如，限制值传感器，用于测量容器中的液位，并在到达最高液位时报告) 作出响应，则当输出该信号时需要处理的子程序可位于事件驱动处理型 OB 上。

线性编程与结构化编程

可以在 OB1 中写入整个用户程序(线性编程)。只有在给 S7-300 CPU 编写简单程序、并要求极少存储器时才可行。

将复杂自动化任务分割成反映过程技术功能或可多次处理的小任务，可以更易于控制复杂任务。这些任务以相应的程序段表示，称为块(结构化编程)。



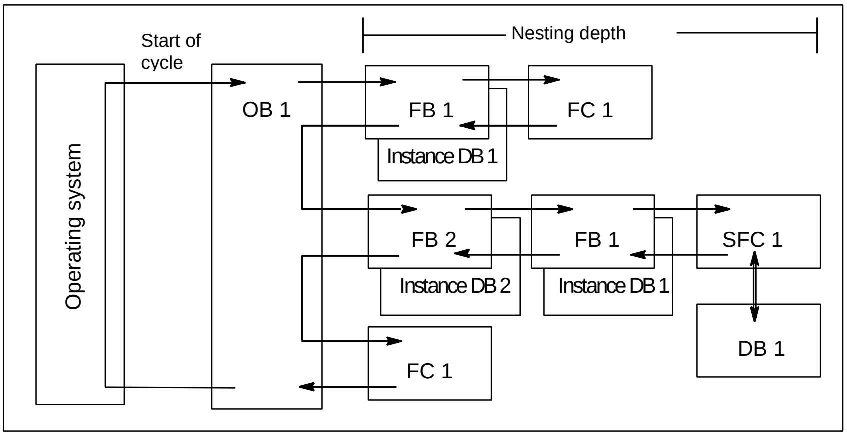
4.2.3 用户程序中的调用体系

要使用户程序正常运行，必须调用构成用户程序的块。这通过特殊的 STEP 7 指令、块调用来完成，而这些指令、块调用只能在逻辑块中编程和启动。

次序和嵌套深度

块调用的次序和嵌套称为体系。可嵌套的块数目(嵌套深度)取决于特定的 CPU。

下图阐述了一个扫描周期内块调用的次序和嵌套深度。



创建块的固定次序如下：

- 从上到下创建块，因此可以从块的顶行开始。
- 调用的每个块必须已经存在，即在一行块内，创建块的次序为从右到左。
- 最后要创建的块是 OB1。

在图中所示的实例中应用这些规则，则可以按下列顺序创建块：

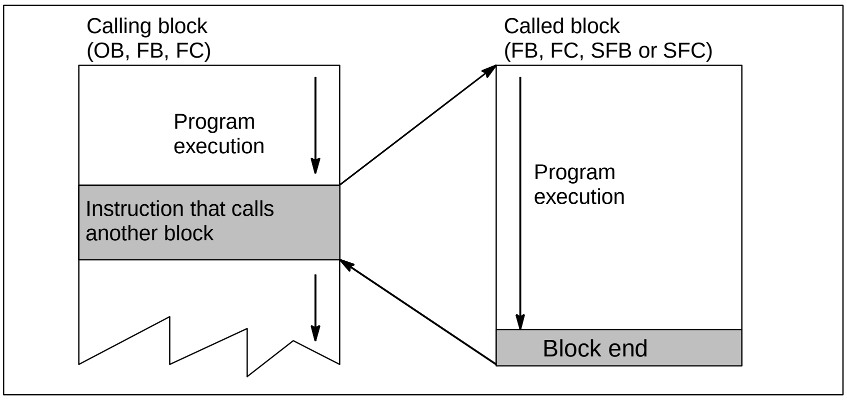
FC1 > FB1 +背景数据块 1 > DB1 > SFC1 > FB2 +背景数据块 2 > OB1

注意

如果嵌套太深(嵌套层次过多)，则本地数据堆栈可能上溢(参见本地数据堆栈)。

块调用

下图显示了在用户程序内块调用的顺序。该程序调用第二个块，然后完全执行该块的指令。一旦执行第二个块或调用块后，在块调用后的指令处继续执行发出调用命令的中断块。



对块进行编程之前，必须指定该程序将使用哪些数据，即，必须声明块变量。

注意

- 必须给每个块调用描述OUT参数。
- 当执行冷重启时，操作系统将SFB3“TP”背景复位。如果在冷重启后，初始化该SFB的背景，那么必须通过OB100以PT = 0毫秒调用SFB的相关背景。例如，可通过在包含该SFB背景的块中执行初始化程序完成该操作。

4.2.4 块类型

4.2.4.1 用于循环程序处理的组织块(OB1)

循环程序处理是在可编程逻辑控制器上执行程序的“正常”类型。操作系统循环调用 OB1，并通过该调用，启动循环执行用户程序。

循环程序处理的顺序

下表显示了循环程序处理的各个阶段：

步骤	98年10月之前的CPU中的顺序	98年10月之后的CPU中的顺序
1	操作系统启动周期监视时间。	操作系统启动周期监视时间。
2	CPU读取输入模块的输入状态，并更新输入的过程映像表。	CPU将来自输出过程映像表的值写入到输出模块。
3	CPU处理用户程序并执行程序所包含的指令。	CPU读取输入模块的输入状态，并更新输入的过程映像表。
4	CPU将来自输出过程映像表的值写入到输出模块。	CPU处理用户程序并执行程序所包含的指令。
5	在周期结束时，操作系统执行未决的任务，例如下载和删除块、接收和发送全局数据。	在周期结束时，操作系统执行未决的任务，例如下载和删除块、接收和发送全局数据。
6	最后，CPU返回周期开始，并重新开始周期监视时间。	最后，CPU返回周期开始，并重新开始周期监视时间。.

过程映像

为了在循环程序处理期间，CPU 具有一致的过程信号映像，CPU 不是直接在 I/O 模块上寻址输入(I)和输出(Q)地址区，而是寻址包含输入和输出映像的 CPU 的内部存储区。

编程循环程序处理

使用 STEP 7，可通过在 OB1 以及 OB1 调用的块中写入用户程序来编程循环程序处理。

一旦无错完成启动程序，就开始循环程序处理。

中断

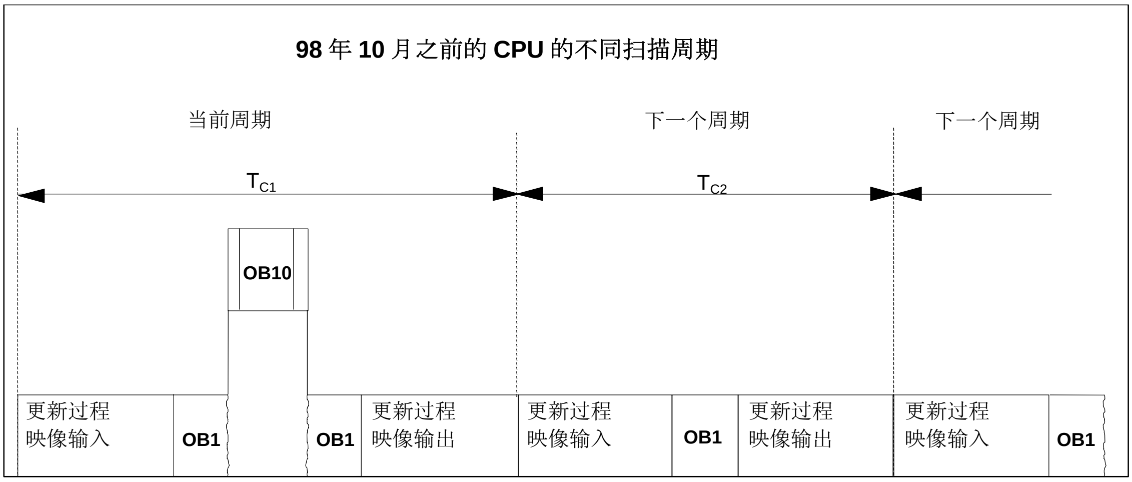
循环程序处理可有下列中断：

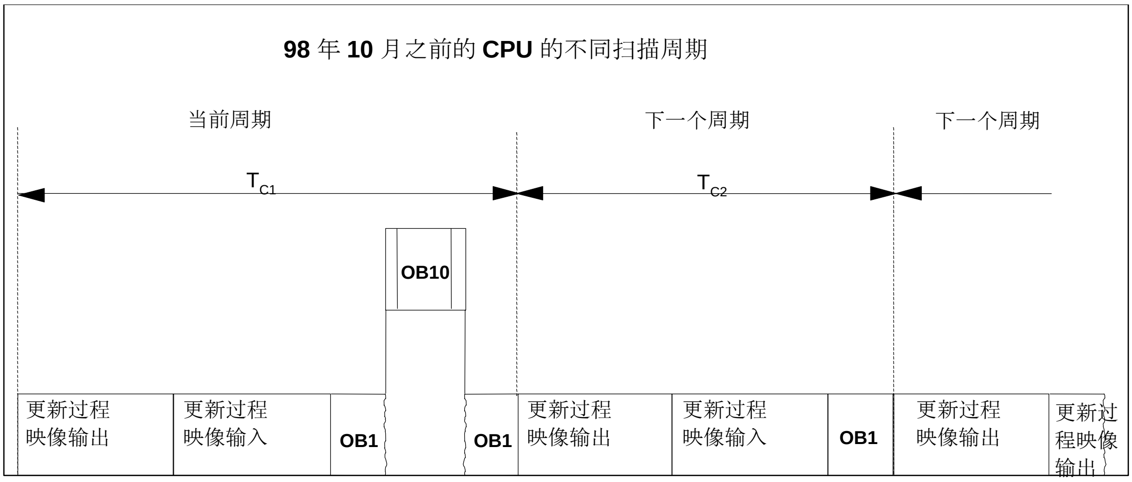
- 中断
- STOP 命令(编程设备、SFC46 STP、SFB20 STOP 上的模式选择器、菜单选项)
- 断电
- 发生故障或程序错误

扫描周期

扫描周期是操作系统运行循环程序以及中断该循环(例如，执行其它组织块)和系统活动(例如，更新过程映像)的所有程序段所需的时间。监视该时间。

每个周期中的扫描时间(TC)均不相同。下图显示了 98 年 10 月之前的 CPU 和 98 年 10 月之后的 CPU 的不同扫描周期(TC1≠TC2)：





在当前周期中，OB1 由时间中断来中断。

周期监视时间

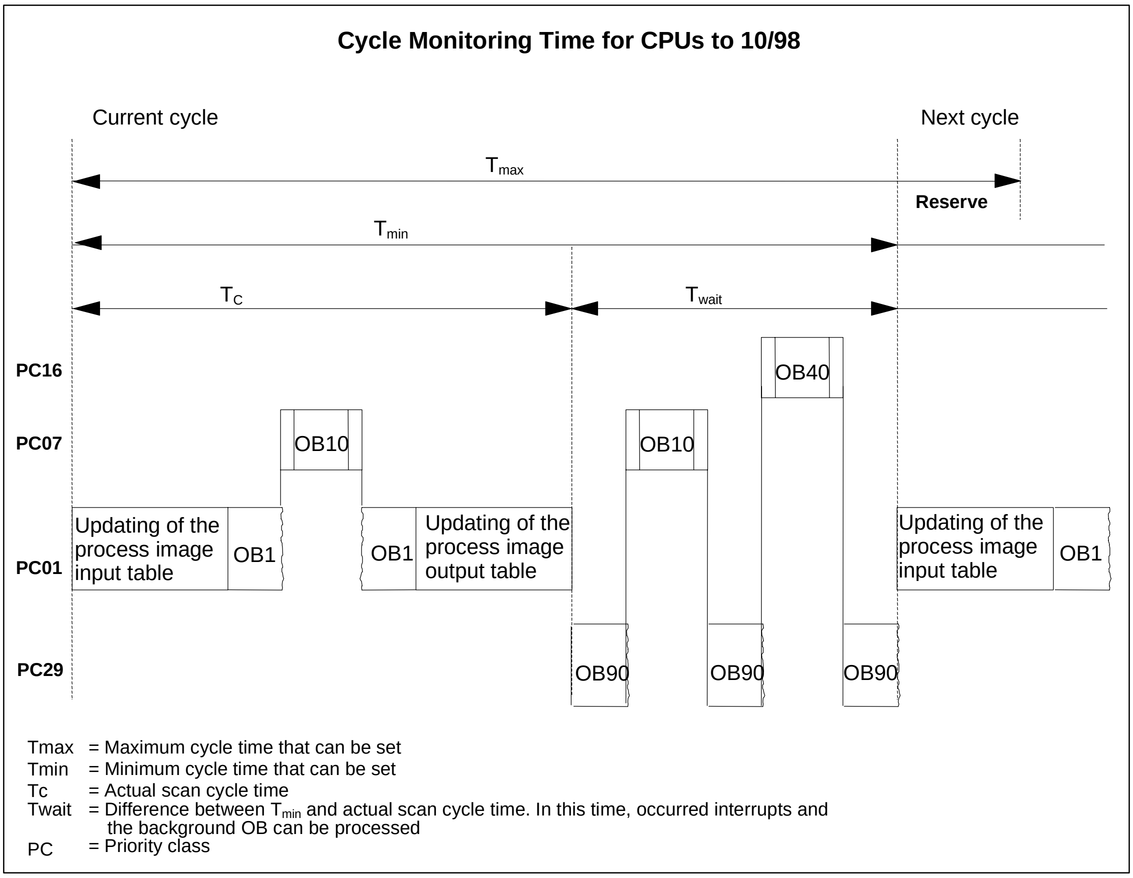
通过STEP 7，可以修改缺省的最大周期监视时间。如果超过该时间，CPU要么进入STOP模式，要么调用OB80，在该OB80中，可以指定CPU如何响应该错误。

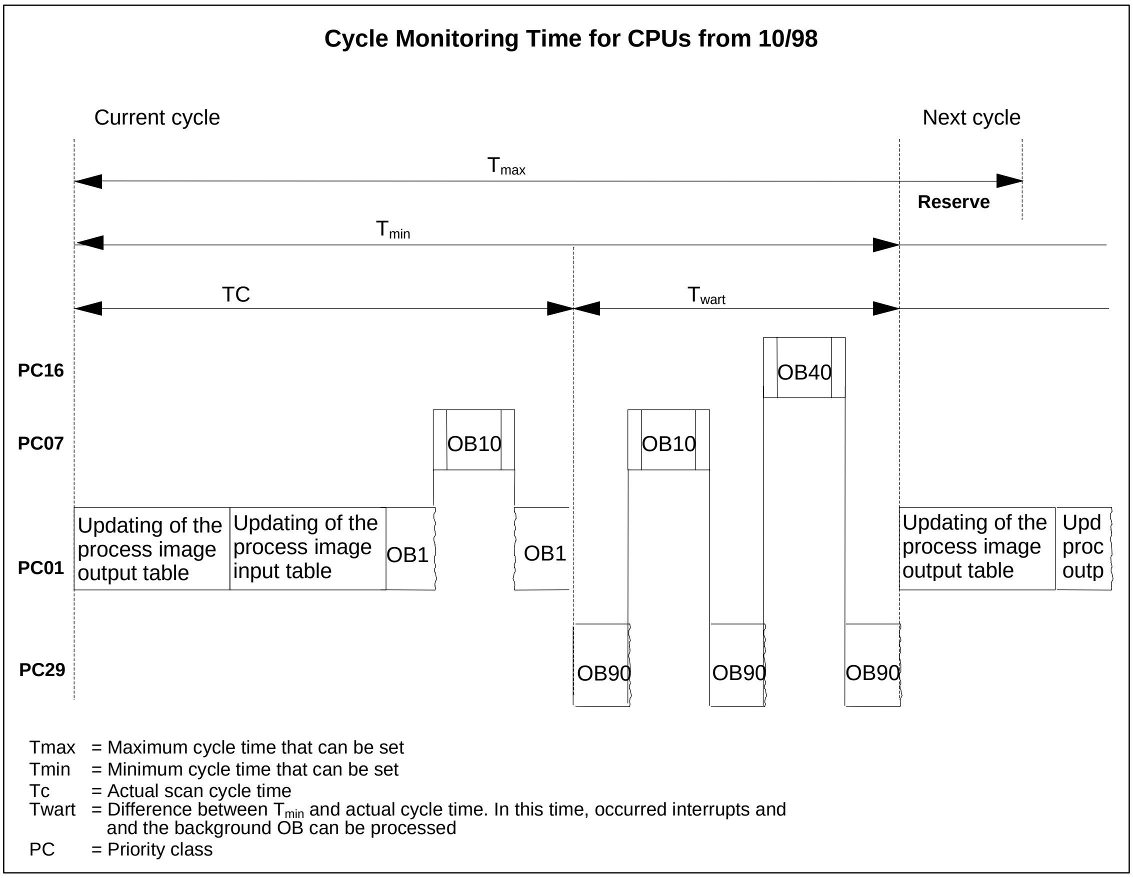
最小周期

通过STEP 7，可以给S7-400 CPU和CPU 318设置最小周期。这在下列情况下非常有用：

- 在 OB1 (主程序扫描)中开始程序执行的时间间隔始终应该相同时或
- 周期太短时，无需经常更新过程映像表。

下图显示了在 98 年 10 月之前的 CPU 以及 98 年 10 月之后的 CPU 中进行程序处理的周期监视时间功能。





更新过程映像

在CPU处理循环程序期间，自动更新过程映像。对于S7-400 CPU和CPU 318，如果希望执行下列动作，那么可以取消选择更新过程映像：

- 直接访问 I/O 或
- 使用系统功能 SFC26 UPDAT_PI 和 SFC27 UPDAT_PO，在程序的不同处更新一个或多个过程映像输入或输出部分。

通讯负载

可以使用“因通讯引起的扫描周期负载”CPU 参数，在给定的框架内控制总是增大扫描周期的通讯过程的持续时间。通讯过程实例包括，通过 MPI 将数据传送至另一个 CPU，或通过编程设备装载块。

使用一个编程设备的测试功能仅受该参数影响。不过，可显著增大扫描周期。在过程模式中，可以限制测试功能的时间设置(仅适用于 S7-300)。

该参数如何工作

CPU操作系统总是按照组态好的百分比为通讯提供CPU处理能力(时间片技术)。如果通讯不需要该处理能力，那么它可用于剩余的处理过程。

对实际扫描周期的影响

没有附加异步事件时，OB1 扫描周期可由一个因素延长，根据下列公式进行计算：

$$\frac{100}{100 - \text{"Scan cycle load from communication (\%)"}}$$

实例 1 (无附加异步事件)：

当将由通讯带给周期的负载设置为 50%时，OB1 扫描周期翻倍。

同时，OB1 扫描周期也受异步事件的影响(例如硬件中断或循环中断)。从统计学角度看，由于因通讯部分延长了扫描周期，在 OB1 扫描周期内会发生更多异步事件。这使得 OB1 扫描周期额外增大。该增量取决于每个 OB1 扫描周期中发生的事件数目以及事件处理的持续时间。

实例 2 (考虑附加异步事件)：

对于一个 500 毫秒的纯 OB1 执行时间，50%的通讯负载将使实际扫描周期高达 1000 毫秒 (假定 CPU 始终有足够的通讯作业要处理)。如果与此同时，每隔 100 毫秒执行一次处理时间为 20 毫秒的循环中断，那么在没有通讯负载时，该循环中断将使扫描周期延长共 5*20 毫秒 = 100 毫秒。即，实际扫描周期将为 600 毫秒。由于循环中断也中断通讯，因此在有 50%的通讯负载时，将扫描周期延长 10 * 20 毫秒。即，此时，实际扫描周期将达 1200 毫秒而不是 1000 毫秒。

注意

- 在系统运行时，请检查因改变“因通讯引起的扫描周期负载”参数而产生的影响。
- 设置最小扫描周期时，必须考虑通讯负载；否则，将发生时间错误。

建议

- 可能时，应用缺省值。
- 只有在 CPU 主要用于通讯目的，并且用户程序对时间要求不是很严格时才增大该值。
- 在所有其它情况下，只能减小该值。
- 设置过程模式(仅适用于 S7-300)，并限制测试功能所需的时间。

4.2.4.2 功能(FC)

功能(FC)属于个人自己编程的块。功能是一种“不带存储器”的逻辑块。属于 FC 的临时变量保存在本地数据堆栈中。执行 FC 时，该数据将丢失。为永久保存该数据，功能也可使用共享数据块。

由于 FC 本身没有存储器，因此，必须始终给它指定实际参数。不能给 FC 的本地数据分配初始值。

应用

FC包含由另一个逻辑块调用该FC时，始终执行的程序段。可使用下列功能：

- 将功能值返回调用块(例如：算术功能)
- 执行技术功能(例如：具有位逻辑操作的单个控制功能)。

将实际参数分配给形式参数

形式参数是“实际”参数的哑元。调用该功能时，实际参数将替换形式参数。必须始终将实际参数分配给FC的形式参数(例如，实际参数“I 3.6”分配给形式参数“Start”)。FC所使用的输入、输出以及输入/输出参数作为指针保存到调用FC的逻辑块的实际参数中。

FC和FB输出参数之间的重要区别

在功能块(FB)中，访问参数时，使用背景数据块中的实际参数副本。如果调用 FB 时，没有传送输入参数或没有写访问输出参数，那么将使用原先保存在背景数据块(背景数据块 = FB 存储器)中的值。

但功能(FC)没有存储器因此，与 FB 相反，将形式参数分配给这些 FC 不是可选，而是必须的。通过地址(指针跨过区域边界指向目标)访问 FC 参数。当数据区(数据块)地址或调用块的局部变量用作实际参数时，实际参数的副本将临时保存到用于传送参数的调用块的本地数据区中。

警告

此时，如果没有数据写入到FC中的OUTPUT参数，那么该块可能输出随机值!

由于没有将保留给副本的调用块的本地数据区分配给OUTPUT参数，因此没有数据写入到该区域。由此，该区保持不变，例如，由于缺省情况下本地数据没有自动设置为“0”，所以将输出在该位置保存的随机值。

因此，请遵守下列几点：

- 如有可能，请初始化 OUTPUT 参数。
- 根据 RLO 将指令进行置位和复位。当这些指令用于判定 OUTPUT 参数值时，如果上一个逻辑操作(RLO)的结果为 0，那么不产生值。
- 始终确保将数据写入到 OUTPUT 参数中，而与块中的程序路径无关。尤其注意跳转到 LAD 和 FBD 中的 ENO 输出以及到 BEC (块结束条件)的跳转指令，以及对 MCR (主站控制继电器)指令的影响。

注意

虽然FB的OUTPUT参数或FC和FB的INOUT参数不会输出随机值(即使没有数据写入到该参数中，也保持原输出值或作为输出值的输入值)，仍然应该遵守上述各点注意事项，避免意外处理“原”值。

4.2.4.3 功能块(FB)

功能块(FB)属于个人自己编程的块。功能块是一种“带存储器”的块。分配数据块作为其存储器(背景数据块)。传送到 FB 的参数和静态变量保存在背景数据块中。临时变量则保存在本地数据堆栈中。

执行完 FB 时，不会丢失背景数据块中保存的数据。但执行完 FB 时，会丢失保存在本地数据堆栈中的数据。

注意

操作FB时，为避免出现错误，请阅读附录中的参数传送时允许的数据类型。

应用

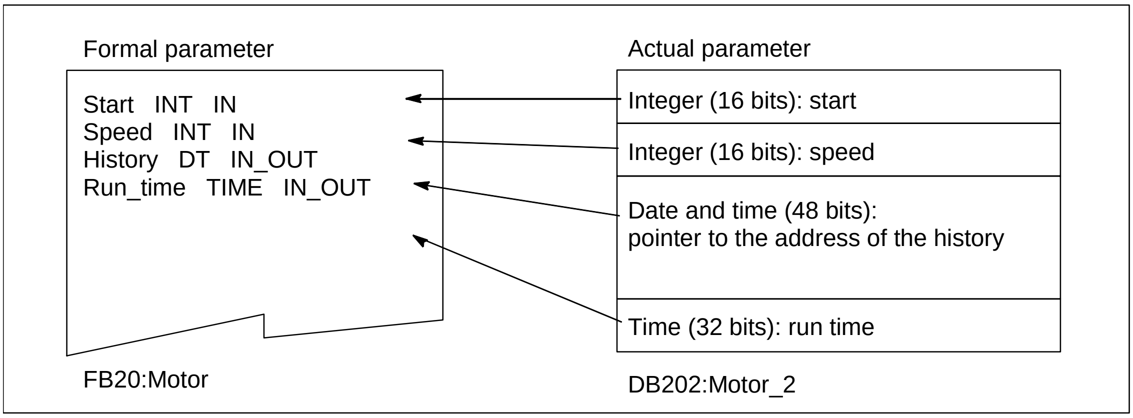
FB包含由其它逻辑块调用FB时始终执行的程序。功能块可以使频繁发生的复杂功能更易于编程。

功能块和背景数据块

背景数据块可分配给传送参数的每个功能块调用。

通过调用一个 FB 的多重背景，可以通过一个 FB 控制多个设备。例如，用于电机类型的 FB 可以通过给不同电机使用不同的背景数据集来控制各个电机。每个电机的数据(例如，速度、斜坡、累积操作时间等)可以保存在一个或多个背景数据块中。

下图显示了使用保存在背景数据块中的实际参数的 FB 的形式参数。



数据类型FB的变量

如果将用户程序结构化，以便FB能包含现存功能块的调用，那么可以在调用FB的变量声明表中包含作为数据类型FB的静态变量调用的FB。该技术允许在一个背景数据块(多重背景)中嵌套变量和集中背景数据。

将实际参数分配给形式参数

通常没有必要在STEP 7中将实际参数分配给FB的形式参数。但也有例外情况。但下列场合中必须分配实际参数：

- 复杂数据类型的输入/输出参数(例如， STRING、ARRAY、DATE_AND_TIME)
- 所有参数类型(例如 TIMER、COUNTER 或 POINTER)

STEP 7 按如下所述将实际参数分配给 FB 的形式参数：

- *在调用声明中指定实际参数时：*FB指令使用所提供的实际参数。
- *没有在调用声明中指定实际参数时：*FB指令使用保存在背景数据块中的值。

下表显示了必须分配实际参数的 FB 变量。

变量		数据类型	
	基本数据类型	**复杂数据类型**	**参数类型**
输入	不需要参数	不需要参数	需要实际参数
输出	不需要参数	不需要参数	需要实际参数
输入/输出	不需要参数	需要实际参数	–

将初始值分配给形式参数

可以在 FB 的声明部分将初始值分配给形式参数。这些值写入到与 FB 有关的背景数据块中。

如果在调用声明中没有将实际参数分配给形式参数，则 STEP 7 将使用保存在背景数据块中的值。这些值也可以是在 FB 变量声明表中输入的初始值。

下表显示了可以分配初始值的变量。由于执行块后，临时数据已丢失，因此不能给它们分配任何值。

变量		数据类型	
	基本数据类型	**复杂数据类型**	**参数类型**
输入	允许初始值	允许初始值	–
输出	允许初始值	允许初始值	–
输入/输出	允许初始值	–	–
静态	允许初始值	允许初始值	–
临时	–	–	–

4.2.4.4 背景数据块

将背景数据块分配给传送参数的每个功能块调用。实际参数和FB的静态数据保存在背景数据块中。在FB中声明的变量确定背景数据块的结构。背景即指功能块调用。例如，如果在S7 用户程序中调用一个功能块5次，那么有5个该块的背景。

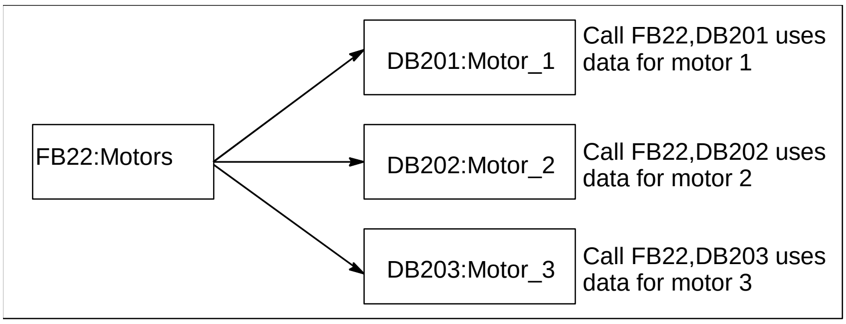
创建背景数据块

创建背景数据块之前，必须存在相应的FB。创建背景数据块时指定FB的编号。

每个单独背景都有一个背景数据块

如果将多个背景数据块分配给控制电机的功能块(FB)，那么可以使用该 FB 来控制不同电机。

每个特定电机的数据(例如，速度、起动时间、总操作时间)保存 在不同的数据块中。调用时与 FB 有关的 DB 决定控制哪台电机。通过该技术，只需一个功能块就可用于多个电机(参见下图)。

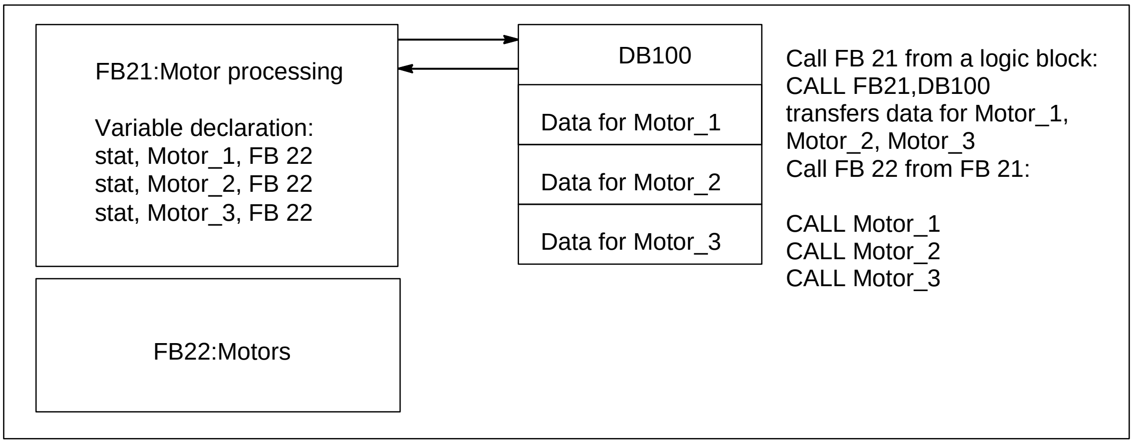


一个背景数据块可用于一个FB的多个背景(多重背景)

可以在一个背景数据块中同时给多个电机传送背景数据。为此，必须在另一个 FB 中编程调用电机控制器，并在调用 FB 的声明部分给单个背景以数据类型 FB 声明静态变量。

给一个 FB 的多个背景使用一个背景数据块，可以节省存储器，优化使用数据块。

在下图中，调用 FB 为 FB21“电机处理”，其变量数据类型为 FB22，背景由 Motor_1、Motor_2 和 Motor_3 标识。



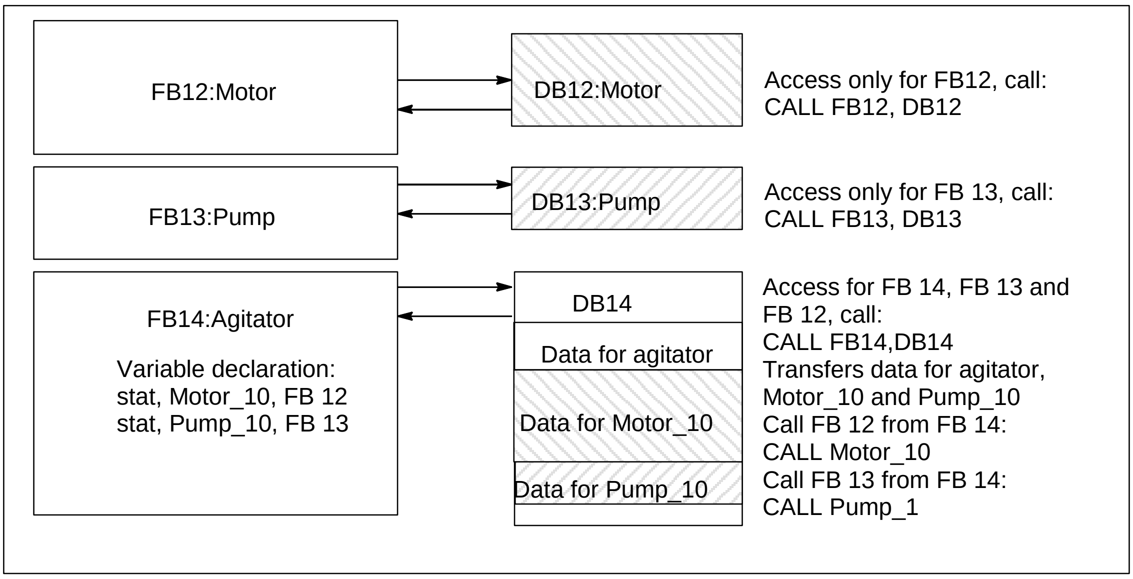
在该实例中，FB22 不需要本身的背景数据块，因为其背景数据保存在调用 FB 的背景数据块中。

一个背景数据块就可用于不同FB的多个背景(多重背景)

在功能块中，可以调用其它已存在的 FB 背景。可以将所要求的背景数据分配给调用 FB 的背景数据块，表示此时不需要给已调用的 FB 提供任何附加数据块。

对于这些在一个背景数据块中的多重背景，必须在调用功能块的声明部分给每个单独背景以已调用功能块的数据类型声明静态变量。因此，功能块内的调用不要求背景数据块，只需要变量的符号名。

在下图的实例中，已分配的背景数据块保存在通用背景数据块中。



4.2.4.5 共享数据块(DB)

与逻辑块相反，数据块不包含 STEP 7 指令。它们用来存储用户数据，即，数据块包含用户程序使用的变量数据。共享数据块则用来存储可由所有其它块访问的用户数据。

共享数据块的大小各不不同。请参见 CPU 说明，了解数据块大小的最大值。

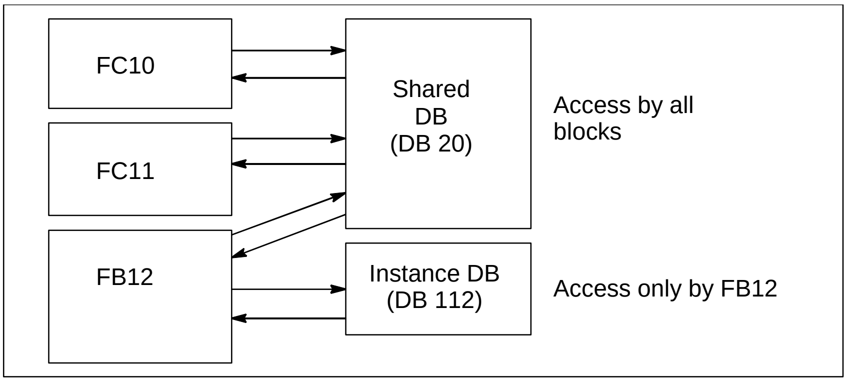
可以任何方式构造共享数据块，满足特定要求。

用户程序中的共享数据块

如果调用逻辑块(FC、FB 或 OB)，它可在本地数据区(L 堆栈) 临时占用空间。除了该本地数据区外，逻辑块还可以 DB 的形式打开存储器区。与本地数据区的数据相反，DB 中的数据在关闭 DB 时，即执行相应的逻辑块后，不会被删除。

每个 FB、FC 或 OB 都可以从共享的 DB 中读取数据或将数据写入到共享 DB。退出 DB 后，该数据仍然保存在 DB 中。

可同时打开共享数据块和背景数据块。下图显示了访问数据块的不同方法。



4.2.4.6 系统功能块(SFB)和系统功能(SFC)

预编程块

没有必要对每个功能都自己编程。S7 CPU 提供可以在用户程序中调用的预编程块。

可在系统块和系统功能的参考帮助中获得更多信息(跳转到关于块、系统属性的语言描述与帮助)。

系统功能块

系统功能块(SFB)是集成在 S7 CPU 中的功能块。SFB 是操作系统的一部分，不作为程序的一部分而被加载。同 FB 一样，SFB 也是“具有存储器”的块。必须给 SFB 创建背景数据块，然后将它们作为程序的一部分下载到 CPU 中。

S7 CPU 给 SFB 提供下列功能：

- 通过已组态的连接进行通讯
- 集成的特殊功能(例如，CPU 312 IFM 和 CPU 314 IFM 中的 SFB29“HS_COUNT”)。

系统功能

系统功能是集成在S7 CPU中的预编程功能。可以在程序中调用SFC。SFC属于操作系统，不能作为程序的一部分而被加载。同FC一样， SFC也是“具有存储器”的块。

S7 CPU 给 SFC 提供下列功能：

- 复制功能和块功能
- 检查程序
- 处理时钟和运行仪表
- 传送数据集
- 在多值计算模式中将事件从一个 CPU 传送到其它 CPU 中
- 处理日历和延时中断
- 处理同步错误、中断和异步错误
- 关于静态和动态系统数据的信息，例如，诊断
- 过程映像更新以及位域处理
- 寻址模块
- 分布式 I/O
- 全局数据通讯
- 通过未组态的连接进行通讯
- 产生与块有关的消息

附加信息

欲知 SFB 和 SFC 的更多详细信息，请参见“S7-300 和 S7-400 系统软件，系统和标准功能”参考手册。“S7-300 可编程控制器，硬件和安装手册”和“S7-400、M7-400 可编程控制器模块规范参考手册”说明提供哪些 SFB 和 SFC。

4.2.5 用于中断驱动程序处理的组织块

4.2.5.1 用于中断驱动程序处理的组织块

通过提供中断 OB，S7 CPU 允许：

- 程序段在特定的时间或间隔内执行(时间驱动)
- 程序可以从进程里响应外部信号。

循环的用户程序不需要询问中断事件是否已发生。如果中断确已发生，操作系统将确保用户程序在中断 OB 中执行，因而可编程控制器将按已编写的动作对中断作出反应。

中断类型和应用

下表给出了如何使用不同类型的中断。

中断类型	中断OB	应用实例
时间中断	OB10至OB17	计算在每个轮班结束时进入混合过程的总流量
延时中断	OB20到OB23	控制一个必须在电机关闭后继续运转20秒钟的风扇
循环中断	OB30到OB38	为闭环控制系统扫描信号电平
硬件中断	OB40到OB47	发信号指示罐已达到最大的液位

4.2.5.2 日历中断组织块(OB10至OB17)

S7 CPU 提供了时间中断 OB，可以在指定的日期或特定的间隔来执行。

时间中断可以如下来触发：

- 在特定的时间执行一次(使用日期以独立的形式指定)
- 周期性地，指定启动时间和中断重复的时间间隔(例如，每分钟、每小时、每天)。

时间中断的规则

只有当中断分配了参数以及用户程序中存在有相应的组织块时，才能执行时间中断。否则，将在诊断缓冲区中输入一条出错消息，并执行非同步的错误处理(OB80，参见错误处理组织块(OB70 至 OB87 / OB121 至 OB122))。

周期性的时间中断必须对应于实际日期。从1月31日开始，每月重复OB10是做不到的。这种情况下，OB只会在确有31天的月份启动(也就是说，不会在二月、四月、六月等月份启动)。

在启动(重新启动(暖启动)或热启动)期间激活的时间中断，只有当启动完成之后才执行。

由参数分配取消选择的时间中断 OB 不能启动。CPU 将识别编程错误，并切换到 STOP 模式。

重新启动(暖启动)之后，必须重新设置时间中断(例如，在启动程序中使用 SFC30 ACT_TINT)。

启动时间中断

要让 CPU 来启动时间中断，必须先对其设置，然后将其激活。有三种方式来启动中断：

- 使用 STEP 7 给时间中断分配合适的参数进行自动启动(参数块“时间中断”)
- 在用户程序中使用 SFC28 SET_TINT 和 SFC30 ACT_TINT 设置并激活时间中断
- 使用 STEP 7 分配参数设置时间中断，并在用户程序中使用 SFC30 ACT_TINT 激活时间中断。

查询时间中断

要查询哪些时间中断作了设置以及它们设置为何时发生，可以采用下列方法：

- 调用 SFC31 QRY_TINT
- 请求系统状态列表的“中断状态”列表。

取消激活时间中断

可以使用SFC29 CAN_TINT取消激活还未被执行的时间中断。被取消激活的时间中断可以用SFC28 SET_TINT重新设置，用SFC30 ACT_TINT激活。

时间中断OB的优先级

所有八个时间中断OB都具有缺省的相同优先级(2)，因此只能按照它们的启动事件发生的顺序进行处理。然而，也可以选择合适的参数来改变优先级。

改变设置时间

可以如下改变中断的时间设置：

- 通过时钟主站同步主站和从站的时间。
- 在用户程序中调用 SFC0 SET_CLK 来设置新的时间。

改变时间的执行结果

下表给出了时间改变后，时间中断如何反应。

如果...	那么...
如果将时间前移，跳过了一个或多个时间中断，	启动OB80，将那些被跳过的时间中断输入OB80的启动信息中。
还没有在OB80中取消激活跳过的时间中断，	跳过的时间中断将不再被执行。
还没有在OB80中取消激活跳过的时间中断，	执行跳过的第一个时间中断，而其它跳过的则被忽略。
将时间退后，时间中断的启动事件将再次发生，	不重复执行日时钟中断。

4.2.5.3 延时中断组织块(OB20至OB23)

S7 CPU 提供了延时 OB，您可用来对部分用户程序的延迟执行进行编程。

延时中断的规则

只有当 CPU 程序中存在相应的组织块时才能执行延时中断。否则，将在诊断缓冲区中输入一条出错消息，并执行异步错误处理(OB80，参见 错误处理组织块(OB70 至 OB87 / OB121 至 OB122))。

没有通过所分配参数选择的延时 OB 将不能启动。CPU 将识别编程错误，并切换到 STOP 模式。

当超过 SFC32 SRT_DINT 中所指定的延时时，将触发延时中断。

启动延时中断

为启动延时中断，您必须指定 SFC32 中的延时，之后，将调用相应的延时中断 OB 。请参见“S7-300 可编程控制器，硬件及安装手册”和“S7-400、M7-400 可编程控制器模块技术规范说明参考手册”中延时的最大允许长度。

延时中断OB的优先级

延时中断 OB 的缺省优先级是优先级 3 至 6。您可通过参数分配改变优先级。

4.2.5.4 循环中断组织块(OB30-OB38)

S7 CPU 提供了循环中断 OB，可以以一定的间隔中断循环程序的进程。

循环中断每隔一段时间触发。间隔启动的时间从 STOP 到 RUN 的模式转换期间。

循环中断的规则

当指定间隔时，确保在单个循环中断的启动事件之间有足够的时间来处理循环中断本身。

如果分配参数时取消选定循环中断 OB，将不能再启动它们。CPU 将识别编程错误，并切换到 STOP 模式。

启动循环中断

要启动一个循环中断，必须使用 STEP 7 在循环中断参数块中指定时间间隔。时间间隔总是基本时钟值 1 毫秒的整数倍。

时间间隔 = n x 基本时钟频率 1 毫秒

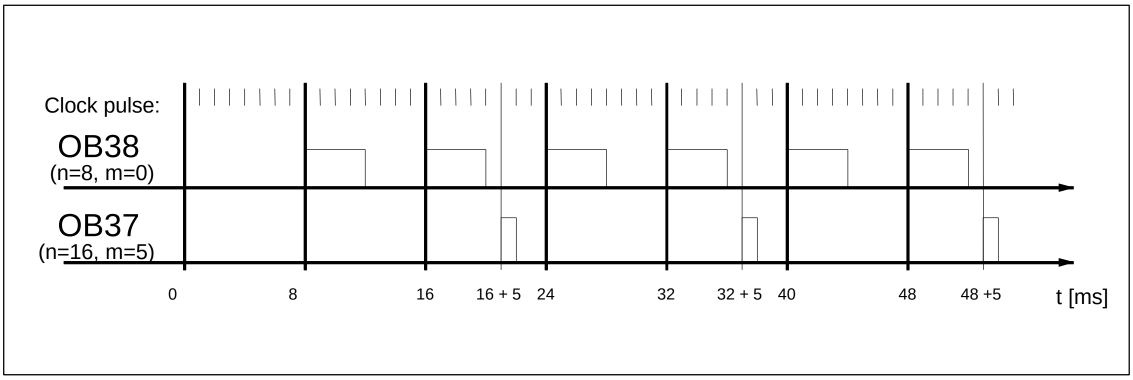
九个可用的循环中断 OB 每个都有缺省的时间间隔(参见下表)。当装载了为缺省间隔所分配的循环中断 OB 后，它便能生效。然而，也可以通过分配参数来改变缺省值。关于上限可参见“S7-300 可编程控制器、硬件及安装手册”和“S7-400、M7-400 可编程控制器模块技术规格参考手册”。

循环中断的相位偏移量

为了避免不同循环中断 OB 的循环中断在同一时间点启动，从而引起可能的时间错误(周期超出)，可以为其指定相位偏移量。相位偏移量能确保在时间间隔结束后，延迟一段时间后再执行循环中断。

相位偏移量 = m x 基本时钟频率(其中 0 * m < n)

下表对照不带相位偏移量的循环中断(OB38)，给出了带有相位偏移量的循环中断 OB (OB37)的执行情况。



循环中断OB的优先级

下表给出了循环中断OB的缺省时间间隔和优先级等级。可以分配参数来改变时间间隔和优先级。

循环中断OB	间隔， 单位：毫秒	优先级
OB30	5000	7
OB31	2000	8
OB32	1000	9
OB33	500	10
OB34	200	11
OB35	100	12
OB36	50	13
OB37	20	14
OB38	10	15

4.2.5.5 硬件中断组织块(OB40至OB47)

S7 CPU 提供了对模块(例如，信号模块(SM)、通讯处理器(CP)、功能模块(FM))信号进行响应的硬件中断 OB。使用 STEP 7，您可决定来自可组态数字模块或模拟模块的哪个信号将启动 OB。对于 CP 和 FM，可使用相应的参数分配对话框。

当具有硬件中断功能与已激活硬件中断的信号模块把所接收的过程信号传递给 CPU 时，或当 CPU 的功能模块发出一个中断信号时，都将触发硬件中断。

硬件中断的规则

只有当 CPU 程序中存在相应的组织块时才能执行硬件中断。否则，将在诊断缓冲区中输入一条出错消息，并执行异步错误处理(OB80，参见 错误处理组织块 (OB70 至 OB87 / OB121 至 OB122))。

如果您在参数分配中没有选择硬件中断 OB，则这些操作均不能启动。CPU 将识别编程错误，并切换到 STOP 模式。

为具有硬件中断功能的信号模块分配参数

具有硬件中断功能的信号模块，其每个通道都能触发一个硬件中断。因此，在使用STEP 7的具有硬件中断功能的信号模块时，必须在其参数集里进行如下设定：

- 什么将触发硬件中断。
- 将执行哪个硬件中断 OB (执行所有硬件中断的缺省 OB 是 OB40)。

使用STEP 7，可激活功能块硬件中断的生成。可在这些功能块的参数分配对话框中分配其余参数。

硬件中断OB的优先级

硬件中断 OB 的缺省优先级是优先级 16 至 23。您可通过参数分配改变优先级

4.2.5.6 启动组织块(OB100 / OB101 / OB102)

启动类型

有三种完全不同的启动类型：

- 热启动(不适用于 S7-300 和 S7-400H)
- 复位(正常启动)
- 冷启动

下表说明了操作系统在每种启动类型中将调用哪个 OB。

启动类型	相关联的OB
热启动	OB101
复位(正常启动)	OB100
冷启动	OB102

用于启动OB的启动事件

CPU 在下列事件发生之后将执行启动：

- 加电之后
- 在将模式选择开关从 STOP 切换到 RUN/RUN-P 之后
- 在收到来自通讯功能的请求之后
- 在对多值计算模式进行同步之后
- 在链接之后的 H 系统中(仅适用于待机的 CPU)

根据不同的启动事件、所使用的 CPU 及其已设置的参数，将调用相应的启动 OB (OB100、OB101 或 OB102)。

启动程序

通过编写用于复位(正常启动)的组织块 OB100、用于热启动的组织块 OB101 或用于冷启动的组织块 OB102 等的启动程序，您可指定启动 CPU 的条件(RUN 模式的初始值、I/O 模块的启动值等)。

由于没有激活循环监视，因此对启动程序的长度没有任何限制，也没有任何时间限制。在启动程序中不能完成由时间驱动或由中断驱动的执行。在启动期间，所有数字输出的信号状态均为0。

手动复位后的启动类型

在 S7-300 CPU 上只能进行手动复位(正常启动)或冷启动(仅适用于 CPU 318-2)。

对于某些 S7-400 CPU，如果根据 STEP 7 进行的参数分配允许的话，您可使用模式选择开关和启动类型开关(CRST/WRST)来进行手动复位。不专门分配参数也能进行手动复位(正常启动)。

自动复位后的启动类型

在 S7-300 CPU 上，只有在加电之后才能进行复位(正常启动)。

对于 S7-400 CPU，您可设定加电之后的自动启动是复位(正常启动)还是热启动。

清除过程映像

当 S7-400 CPU 复位时，剩余的周期将继续执行，并且在缺省状态下，将清除过程映像输出表。如果希望用户程序在复位之后继续使用原值，则可采取措施避免清除过程映像。

模块存在/类型监视

在各种参数中，您可决定是否对组态表中的模块进行检查，以确保这些模块确实存在并且复位之前模块类型相互匹配。

如果已经激活模块检查，则当组态表与实际组态之间存在差异时，将不启动 CPU。

监视时间

为确保可编程控制器正确启动，您可选择下列监视时间：

- 将参数传送给模块的最大允许时间
- 模块加电后到发出可以操作的信号时的最大允许时间
- 对于 S7-400 CPU，在其间允许热启动时的中断最大时间一旦超出监视时间，

一旦超出监视时间，CPU 既可以切换到 STOP 模式，也可以只进行复位(正常启动)。

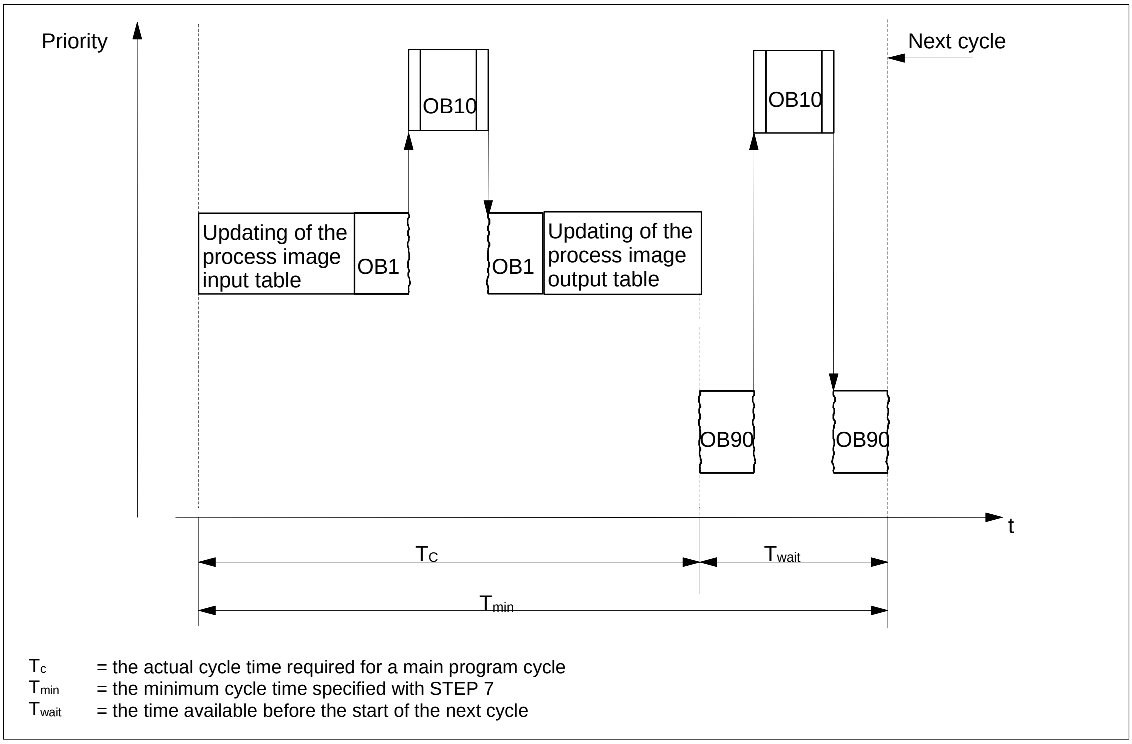
4.2.5.7 背景组织块(OB90)

如果已经用STEP 7设定了最小扫描周期时间，且该时间大于实际扫描周期时间，则CPU将仍然处理，直到程序循环结束。该时间将用于执行背景OB。如果CPU上不存在OB90，则CPU将等待，直到超出设定的最小扫描周期时间。因此，可使用OB90来运行时间要求并不严格的过程，从而避免等待时间。

背景OB的优先级

背景OB具有优先级29，它对应于优先权0.29。因此，这是优先级最低的OB。而且其优先级不能通过重新分配参数来进行修改。

下图给出了一个实例，说明如何处理背景周期、主程序周期以及 OB10 (10/98 至今的 CPU)。



对OB90进行编程

CPU 操作系统并不监视 OB90 的运行时间，因此可对 OB90 中任意长度的循环进行编程。请在编程时注意以下事项，以确保在背景程序中所使用数据的一致性。

- OB90 的复位事件(参见“S7-300 和 S7-400 的系统软件，系统功能和标准功能”参考手册)
- 与 OB90 异步的过程映像更新。

4.2.5.8 错误处理组织块(OB70至OB87 / OB121至OB122)

错误类型

S7 CPU 可检测到错误，并可借助组织块对其进行响应，这些错误可分为两种基本类型：

- 同步错误：可将这些错误分配给用户程序的特定部分。这类错误发生在特定指令的执行期间。如果没有装载相应的同步错误 OB，则 CPU 在发生错误时将切换到 STOP 模式。

- 异步错误：这些错误不能直接分配给正在执行的用户程序。这类错误是优先级错误、可编程逻辑控制器上的故障(例如，故障模块)或冗余错误。如果未装载相应的异步错误 OB，则当出现错误()时，CPU 将转为 STOP 模式
- (例外情况：OB70、OB72、OB81)。

下表列出了可能发生的错误类型，并且已被分为不同种类的错误 OB。

异步错误/冗余错误	同步错误
OB70 I/O 冗余错误(只对H CPU)	OB121 编程错误(例如，尚未装载DB)
OB72 CPU冗余错误(仅适用于H CPU，例如，CPU故障)	OB122 I/O 访问错误(例如，访问一个并不存在的信号模块)
OB80时间错误(例如，超出扫描周期时间)	
OB81电源错误(例如，电池故障)	
OB82诊断错误(例如，输入模块中电路短路)	
OB83拆卸/插入中断(例如，拆卸输入模块)	
OB84 CPU 硬件故障(MPI网络接口处的故障)	
OB85 优先级错误(例如，尚未装载OB)	
OB86机架故障	
OB87通讯错误(例如，全局数据通讯的消息框架ID不正确)	

用于同步错误的OB

同步错误发生在特定指令的执行期间。当这些错误发生时，操作系统将一个条目输入 I 堆栈并启动用于同步错误的 OB。

由于同步错误而调用的错误OB将作为程序的一部分执行，其优先级与检测到错误时正在执行的块相同。因此，OB121和OB122可以访问当发生中断时，累加器和其它寄存器中的数值。您可使用这些数值对错误的产生条件进行响应，然后返回继续处理您的程序(例如，如果在某个模拟量输入模块上发生了一个访问错误，则可使用SFC44 RPL_VAL指定一个OB122的替换值)。但错误OB的本地数据实际上确实占用了该优先级的L堆栈的其余空间。

对于 S7-400 CPU，同步错误 OB 可启动其它同步错误 OB。但对于 S7-300 CPU 则不行。

用于非同步错误的OB

如果CPU的操作系统检测到一个异步错误，则它将启动相应的OB (OB70至OB73以及OB80至OB87)。缺省状态下，用于异步错误的OB的优先级最高，并且，如果所有的异步错误OB都具有同样的优先级，则它们可被其它OB中断。如果同时发生一个以上的异步错误OB，则将按它们发生的顺序对其进行处理。

屏蔽启动事件

使用系统功能(SFC)，可以屏蔽、延迟或禁止多个OB的启动事件。关于这些SFC和组织块的更详细信息，请参见“S7-300和S7-400的系统软件，系统功能与标准功能”参考手册。

错误OB的类型	**SFC**	**SFC的功能**
同步错误OB	SFC36 MSK_FLT	屏蔽单个的同步错误。已屏蔽的错误将不能启动错误OB，也不触发已编程的响应。
	SFC37 DMSK_FLT	未屏蔽的同步错误
异步错误OB	SFC39 DIS_IRT	禁止所有中断和异步错误。被禁止的错误将不启动任何后续CPU周期中的错误OB，也不触发已编程的响应。
	SFC40 EN_IRT	激活中断和异步错误
	SFC41 DIS_AIRT	延迟更高优先级的中断和异步错误，直到OB结束
	SFC42 EN_AIRT	激活更高优先级的中断和异步错误

注意

设计程序结构的基本原理

5 启动和操作

5.1 启动STEP 7

在启动 Windows 时，您将发现一个代表 SIMATIC 管理器的图标，该管理器就是 STEP 7 软件在 Windows 接口上的启动点。

启动 STEP 7 的最快方法就是将光标放置在该图标上，然后双击。随后将打开包含有 SIMATIC 管理器的窗口。从中可访问您为标准软件包和任意选件包所安装的所有功能。

也可通过操作系统任务栏上的“开始”按钮来启动 SIMATIC 管理器。您将发现“Simatic”下的条目。

注意

关于标准Windows操作与选项的更详细信息，请参见Windows用户指南或Windows操作系统的在线帮助。

SIMATIC管理器

SIMATIC管理器是用于组态和编程的基本应用程序。可在SIMATIC管理器中执行下列功能：

- 设置项目
- 配置硬件并为其分配参数
- 组态硬件网络
- 程序块
- 对程序进行调试

已将各种不同功能的访问设计成面向对象的访问，且学习起来，直观、方便。

使用 SIMATIC 管理器进行工作，可采取下列两种方式之一：

- 离线方式，没有连接可编程控制器
- 在线方式，连接了可编程控制器

请注意每种情况下的相关安全注意事项。

如何以此为起点往下操作

您将以“项目”的形式创建自动化任务。在着手开始工作之前，如果仔细阅读下列基本主题，将使工作更容易、更轻松：

- 用户接口
- 一些基本操作步骤
- 在线帮助

5.2 使用缺省启动参数启动STEP 7

从 STEP 7 V5.0 起，在 SIMATIC 管理器中可创建多个符号，并可在调用命令行中设定启动参数。这样，就可将 SIMATIC 管理器放置在这些参数所描述的对象上。从而只需通过双击就能立即跳转到项目中的相应位置。

调用 s7tgtopx.exe 时，可以设定下列启动参数：

/e <完整物理项目路径>

/o <希望定位于其上的对象的逻辑路径>

/h <对象标识号> /onl 或 /off

建立正确参数的最简单方法如下所述：

通过复制和粘贴设置参数

操作过程如下：

1. 在桌面上，给文件s7tgtopx.exe创建一个新链接。
2. 显示属性对话框。
3. 选择“链接”标签。“目标”下的条目现在应按如下扩展。
4. 在SIMATIC管理器中选择所需对象。
5. 使用组合键CTRL+ALT+C将对象复制到剪贴板。
6. 将光标放在“链接”标签中的“目标”条目的末尾。
7. 使用组合键CTRL+V粘贴剪贴板的内容。
8. 使用“确定”进行确认，关闭对话框。

参数实例：

/e F:\SIEMENS\STEP7\S7proj\MyConfig\MyConfig.s7p

/o "1,8:MyConfig\SIMATIC 400(1)\CPU416-1\S7-Program(1)\Blocks\FB1"

/h T00112001;129;T00116001;1;T00116101;16e

关于项目路径结构的注意事项

项目路径是文件系统中的物理路径。因此，不支持 UNC 注释，例如：
F:\SIEMENS\STEP7\S7proj\MyConfig\MyConfig.s7p

完整的逻辑路径具有下列结构：

[视图 ID,在线 ID]:项目名\{对象名\}*\ 对象名

实例：/o 1.8:MyConfig\SIMATIC 400(1)\CPU416-1\S7-Program(1)\Blocks\FB1

关于逻辑路径结构的注意事项

只有使用复制和粘贴功能才能创建完整的逻辑路径与对象 ID。

但也可以指定由用户进行读取的路径。在上述实例中，它可以是：

/o "MyConfig\SIMATIC 400(1)\CPU416-1\S7-Program(1)\Blocks\FB1"。通过添加 /onl 或 /off ，用户可设定路径在在线或离线窗口中是否有效。如果使用复制或粘贴功能，则不需要对其进行设定。

重要事项：如果路径包含有空格，则必须将其放在引号内。

5.3 调用帮助功能

在线帮助

在线帮助系统提供了当前最为有效的信息。使用在线帮助可迅速、快捷地访问各种信息而无需再搜索各种手册。在在线帮助中，可找到下列类型的各种信息：

- 目录：提供了显示帮助信息的各种不同方法
- 上下文关联帮助(F1键)：使用F1键，可访问与刚才使用鼠标所选择的对象有关的信息，或与活动对话框或窗口等有关的信息
- 引言：给出了关于应用程序的使用、主要特点以及功能范围等的简介
- 使用入门：概括了初次使用应用程序时需要执行的基本步骤
- 使用帮助：提供了关于在线帮助中特定信息搜索方法的描述
- 关于：提供了关于应用程序的当前版本信息

通过帮助菜单，也可从每个不同窗口访问与当前对话状况相关的各个主题。

调用在线帮助

可选择下列方式调用在线帮助：

- 选择菜单栏帮助菜单中的菜单命令。
- 单击对话框中的“帮助”按钮。随后将显示关于该对话框的帮助。
- 将光标置于窗口或对话框中需要获得帮助的主题上，然后按 F1 键或选择菜单命令**帮助 > 上下文关联帮助**。
- 使用窗口中的问号符号光标。

在这些访问在线帮助的方式中，我们称后面三种方式为上下文关联帮助。

调用快速帮助

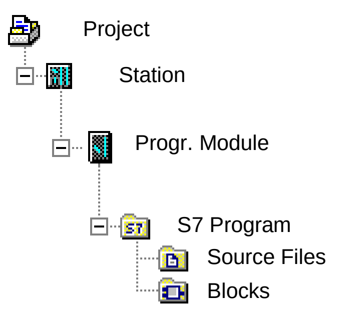
将光标放置在任务栏中的按钮上并让其停留片刻，将显示该按钮的快速帮助。

5.4 对象与对象体系

5.4.1 对象与对象体系

按照与 Windows 资源管理器显示文件夹和文件的目录结构相同的方式，将 STEP 7 中项目和库的对象体系显示在 SIMATIC 管理器中。

下图显示了对象体系的一个实例。



- 项目对象
- 站对象
- 可编程模块对象
- S7/M7程序对象
- 源文件文件夹对象
- 块文件夹对象

对象具有下列功能：

- 对象属性的载体，
- 文件夹，
- 功能的载体(例如，启动特定的应用程序)。

作为属性载体的对象

对象既可以具有功能，也可以具有属性(例如设置)。在选择对象时，可用它来执行下列功能之一：

- 使用菜单命令**编辑 > 打开对象**来编辑对象。
- 使用菜单命令**编辑 > 对象属性**打开对话框，并设置对象特定的选项。

文件夹也可作为属性的载体。

作为文件夹的对象

文件夹(目录)可包含其它文件夹或对象。当打开文件夹时它们均将显示。

作为功能载体的对象

当打开对象时，将显示一个窗口，您可在其中编辑对象。

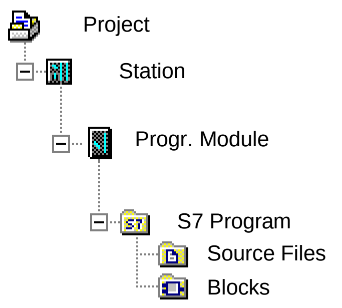
对象或者是一个文件夹，或者是功能的载体。其中的一个例外就是站：它们既是文件夹(对于可编程模块)，也是功能的载体(用于组态硬件)。

- 如果双击一个站，则将显示包含在其中的对象：可编程模块和站组态(作为文件夹的站)。
- 如果使用菜单命令**编辑 > 打开对象**打开一个站，则您可组态该站，为其分配参数(作为功能载体的站)。菜单命令具有与双击“硬件”对象一样的效果。

5.4.2 项目对象

项目代表了自动化解决方案中的所有数据和程序的整体，它位于对象体系的最上层。

在项目视图中的位置



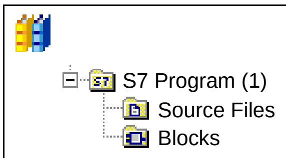
- 项目对象
- 站对象
- 可编程模块对象
- S7/M7程序对象
- 源文件文件夹对象
- 块文件夹对象

符号	对象文件夹	重要功能的选择
	项目	• 创建项目 • 对项目和库进行归档 • 管理多语言文本 • 检查 项目所使用的可选软件包 • 打印项目文档 • 重新排列 • 翻译和编辑与操作员有关的文本 • 插入操作员站对象 • 一个以上的用户编辑项目 • 转换版本1的项目 • 转换版本2的项目 • 设置PG/PC接口

符号	项目层中的对象	重要对象的选择
	站： SIMATIC 300站 SIMATIC 400站	• 插入站 • 站既是对象(项目层)，也是对象文件夹(站层)。其它功能参见站对象
	S7程序 M7程序	• 不带站或CPU的S7/M7程序 • S7/M7程序既是对象(项目层)，也是对象文件夹(程序层)。其它功能参见S7/M7程序对象
	用于启动网络组态工具和设置网络属性的网络。	• 子网和通讯节点的属性 • 概述：全局数据通讯 • 全局数据通讯的组态步骤

5.4.3 库对象

一个库可包含有S7/M7程序，并可用于对块进行存储。库位于对象体系的最上层。

	• 库对象 • S7/M7程序对象 • 源文件文件夹对象 • 块文件夹对象

符号	对象文件夹	重要功能的选择
	库	• 标准库概述 • 使用库进行工作 • 对项目和库进行归档

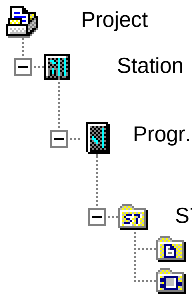
符号	库层中的对象	重要功能的选择

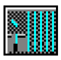
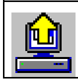
 	S7程序 M7程序	• 插入S7/M7程序 • S7/M7程序既是对象项目层，也是对象文件夹程序层。其它功能参见S7/M7程序对象

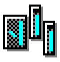
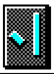
5.4.4 站对象

SIMATIC 300/400 站表示具有一个或多个可编程模块的 S7 硬件组态。

在项目视图中的位置

	• 项目对象 • **站对象** • 可编程模块对象 • S7/M7程序对象 • 源文件文件夹对象 • 块文件夹对象

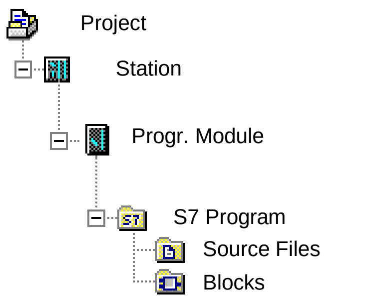
符号	对象文件夹	重要功能的选择
	站	• 插入一个站 • 上传站 • 将组态下载给可编程控制器 • 从站上传组态 • 显示CPU消息和用户自定义的诊断消息 • 组态“系统错误报告”’ • 诊断硬件和显示模块信息 • 显示和改变工作模式 • 显示和设置时间与日期 • 删除装载/工作存储器，并复位CPU
	SIMATIC PC站 (未分配)	• 创建和分配SIMATIC PC站的参数 • 组态SIMATIC PC站的连接 • 上载SIMATIC PC站
	SIMATIC PC站 (已分配)	• 高亮显示将要在网络视图中组态的SIMATIC PC站

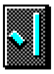
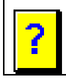
符号	站层中的对象	重要功能的选择
	硬件	• 硬件组态的基本步骤 • 站组态的基本步骤 • 概述：局部组态的参数组态与分配步骤 • 组态DP主站系统的基本步骤 • 组态多值计算操作
	可编程模块	• 可编程模块既是对象(站层)，也是对象文件夹(“可编程模块”层)。其它功能参见可编程模块对象

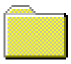
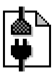
5.4.5 可编程模块对象

可编程模块表示可编程模块(CPUxxx、FMxxx、CPxxx)的参数赋值数据。没有任何保持存储器的模块(例如，CP441)，其系统数据将通过站的 CPU 进行装载。因此，将不为这样的模块分配任何“系统数据”对象，而它们也不在项目体系中显示。

在项目视图中的位置

	• 项目对象 • 站对象 • **可编程模块对象** • S7/M7程序对象 • 源文件文件夹对象 • 块文件夹对象

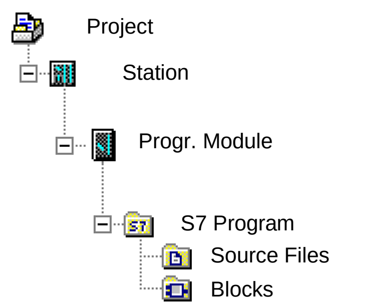
符号	对象文件夹	重要功能的选择
	可编程模块	• 概述：局部组态的参数组态与分配步骤 • 显示CPU消息和用户自定义诊断消息 • 组态“系统错误报告” • 诊断硬件和显示模块信息 • 通过EPROM存储器卡进行下载 • 用于访问可编程控制器的密码保护 • 显示强制值窗口 • 显示和改变工作模式 • 显示和设置时间与日期 • 设置操作特性 • 删除装载/工作存储器，并复位CPU • 在线视图中的诊断符号 • 存储器区的划分 • 在集成的EPROM上保存已下载的块 • 更新可编程逻辑控制器上的操作系统
	代表可编程模块的对象	• 显示使用较新的STEP 7版本组态的模块

符号	“可编程模块”层中的对象	重要功能的选择
  	程序： S7程序 M7程序 程序	• 插入S7/M7程序 • S7/M7程序既是对象(项目层)，也是对象文件夹(程序层)。其它功能参见S7/M7程序对象
	用于定义网络内连接的连接	• 在项目内连网站 • 连接类型和连接伙伴 • 各种连接类型须知 • 输入新的连接 • 组态SIMATIC站中模块的连接

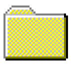
5.4.6 S7/M7程序对象

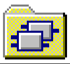
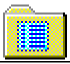
(S7/M7)程序文件夹包含了用于 S7/M7 CPU 模块的软件或用于非 CPU 模块(例如，可编程 CP 或 FM 模块)的软件。

在项目视图中的位置



- 项目对象
- 站对象
- 可编程模块对象
- **S7/M7程序对象**
- 源文件文件夹对象
- 块文件夹对象

符号	对象文件夹	重要功能的选择
	S7程序	• 插入S7-/M7-程序 • 设置地址优先级 • 创建逻辑块的基本步骤 • 分配消息号 • 如何为项目分配和编辑特定用户的诊断消息 • 如何为CPU分配和编辑特定用户的诊断消息 • 翻译和编辑与操作员有关的文本 • 管理多语言文本 • 显示CPU消息和用户自定义诊断消息 处理错误的程序措施
	M7程序	• M7系统的步骤
	程序	• 创建项目中的软件(常规)

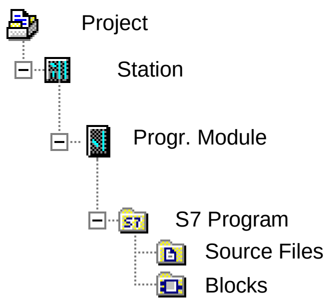
符号	程序层中的对象	重要功能的选择
	源文件文件夹	• 其它功能参见源文件文件夹对象
	块文件夹	• 其它功能参见块文件夹对象
	文本库文件夹	• 用户文本库
	用于为信号和其它变量分配符号的符号表	• 绝对和符号寻址 • 符号表的结构和组件 • 输入共享符号 • 与输入符号有关的常用提示 • 如何为项目分配和编辑与符号相关的消息 • 如何为CPU分配和编辑与符号相关的消息 • 翻译和编辑与操作员有关的文本 • 通过符号表组态操作员监控属性 • 编辑通讯属性 • 导出和导入符号表

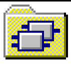
5.4.7 块文件夹对象

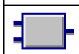
离线视图的块文件夹可包括：逻辑块(OB、FB、FC、SFB、SFC)，数据块(DB)，自定义的数据类型(UDT)和变量表。系统数据对象表示系统数据块。

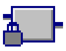
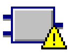
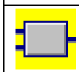
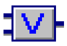
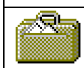
在线视图的块文件夹包括已经下载给可编程控制器的可执行程序部分。

在项目视图中的位置

	• 项目对象 • 站对象 • 可编程模块对象 • S7/M7程序对象 • 源文件文件夹对象 • **块文件夹对象**

符号	对象文件夹	重要功能的选择
	块	• 通过项目管理下载 • 不通过项目管理而进行的下载 • 可用引用数据的概述 • 重新连接 • 比较块 • 翻译和编辑与操作员有关的文本 • 跳转到关于块、系统属性的语言描述与帮助

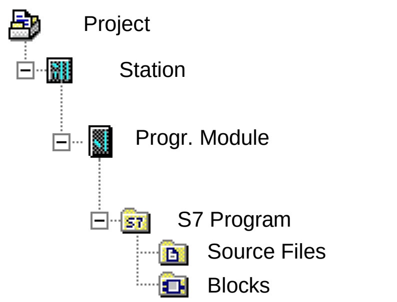
符号	块文件夹中的对象	重要功能的选择
	常用块	• 创建逻辑块的基本步骤 • 创建块 • 关于在STL源文件中编程的基本信息 • 比较块
	组织块(OB)	附加功能： • 数据类型和参数类型的引言 • 下载要求 • 使用程序状态进行测试 • 单步模式/断点测试须知 • 重新连接 • 块帮助
	功能(FC)	附加功能： • 数据类型和参数类型的引言 • 下载要求 • 使用程序状态进行测试 • 单步模式/断点测试须知 • 重新连接 • 块和参数的属性
	功能块(FB)	附加功能： • 数据类型和参数类型的引言 • 使用多重背景 • 下载要求 • 使用编程状态测试 • 关于单步模式/断点中的测试须知 • 重新连接 • 块和参数的属性 • 如何为项目分配和编辑与块相关的消息 • 如何为CPU创建与块相关的消息 • 如何为项目组态PCS 7消息 • 如何为CPU组态PCS 7消息 • 翻译和编辑与操作员有关的文本 • 设置功能块参数的监视/控制属性
	自定义的数据类型(UDT)	• 创建块 • 关于在STL源文件中编程的基本信息 • 数据类型和参数类型的引言 • 使用用户自定义数据类型访问数据 • 块和参数的属性

符号	块文件夹中的对象	重要功能的选择
	DB (全局数据块)	• 数据块的数据视图 • 数据块的声明视图 • 下载要求 • 数据块的程序状态 • 数据类型和参数类型的引言 • 使用多重背景 • 块和参数的属性 • 如何为项目分配和编辑与块相关的消息(仅适用于背景DB) • 如何为CPU分配和编辑与块相关的消息(仅适用于背景DB) • 如何为项目组态PCS7消息(仅适用于背景DB) • 如何为CPU组态PCS7消息(仅适用于背景DB) • 翻译和编辑与操作员相关的文本(仅适用于背景DB)
	系统功能(SFC)	• 下载要求 • 块和参数的属性 • 块帮助
	SFB (系统功能块)	• 下载要求 • 块和参数的属性 • 如何为项目分配和编辑与块相关的消息 • 如何为CPU创建与块相关的消息 • 如何为项目组态PCS7消息 • 如何为CPU组态PCS7消息 • 翻译和编辑与操作员有关的文本 • 块帮助
	具有专业知识保护的块	• STL源中的块属性的定义规则 • 块属性
	具有诊断能力的块	其它附加信息参见S7-PDIAG可选软件包的有关文档。
	已使用F-FBD/-LAD/-STL/-DB编程语言创建了块	其它附加信息参见“S7分布式安全”可选软件包的有关文档。
	变量表(VAT)	• 使用变量表进行监视和修改时的基本步骤 • 使用变量表进行测试的介绍 • 监视变量简介 • 关于对变量进行修改的说明 • 关于对变量进行强制的说明
	系统数据块(SDB)	系统数据块(SDB)只能通过功能间接进行编辑： • 组态硬件介绍 • 子网和通讯节点的属性 • 概述：全局数据通讯 • 分配和编辑与符号相关的消息 • 下载要求

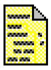
5.4.8 源文件文件夹对象

源文件文件夹包含了文本格式的源程序。

在项目视图中的位置

	• 项目对象 • 站对象 • 可编程模块对象 • S7/M7程序对象 • **源文件文件夹对象** • 块文件夹对象

符号	对象文件夹	重要功能的选择
	源文件文件夹	• 关于在STL源文件中编程的基本信息 • 导出源文件 • 导入源文件

符号	源文件文件夹中的对象	重要功能的选择
	源文件 (例如，STL源文件)	• 关于在STL源文件中编程的基本信息 • 创建STL源文件 • 在STL源文件中插入块模板 • 在STL源文件中插入来自现有块的源代码 • 检查STL源文件中的一致性 • 编译STL源文件 • 生成来自块的STL源文件 • 导出源文件 • 导入源文件
	网络模板	• 使用网络模板进行工作

5.4.9 不带站或**CPU**的**S7/M7**程序

可以在事先没有组态一个 SIMATIC 站时创建程序。这表示最初可独立于希望编程的模块和模块设置工作。

创建S7/M7程序

1. 使用菜单命令**文件 > 打开**，打开相关的项目，或激活项目窗口。
2. 在离线视图项目窗口中选择项目。
3. 根据将使用所创建的程序的可编程控制器，选择以下菜单命令之一：
   插入 > 程序 > S7程序：如果程序在SIMATIC S7设备上运行。
   插入 > 程序 > M7程序：如果程序在SIMATIC M7设备上运行。

S7/M7 程序直接在项目窗口的项目下添加和排列。它包含一个用于块的文件夹和一个空符号表。现在可创建和编程块。

将程序分配给可编程模块

当插入不倚赖于特定模块的程序时，随后可以很容易地将它们分配给模块，方法是使用拖放功能，将这些程序复制或移动到模块符号上。

向库中添加一个程序

如果程序将用于 SIMATIC S7 可编程控制器，并且希望作为“软件库”多次使用，那么还可以在库中插入该程序。不过，测试时，该程序必须直接位于一个项目下，因为这是与可编程控制器建立连接的唯一方式。

访问可编程控制器

选择项目的在线视图。可在包含程序属性的对话框中进行地址设置。

注意

删除站或可编程模块时，会提示是否删除其中所包含的程序。如果选择不删除程序，那么它将作为一个没有站的程序直接连接到项目下。

5.5 用户界面和操作

5.5.1 操作原则

目标：轻松操作

图形用户接口的目的是提供最大程度的、直观的操作方便性。因此，您可以发现日常的工作中熟悉的各种对象，例如，站、模块、程序、块。

您在 STEP 7 下执行的动作包括上述对象的创建、选择和操作。

同基于工具的操作的差别

在使用常规工具开始工作时，您必须做的第一件事就是为特定的解决方案选择合适工具，然后调用该工具。

面向对象操作的基本步骤是选择一个对象，然后将其打开以进行编辑。

面向对象的操作不需要特殊的指令语法的知识。在 GUI 上，通过菜单命令或鼠标点击打开的图标代表了对象。

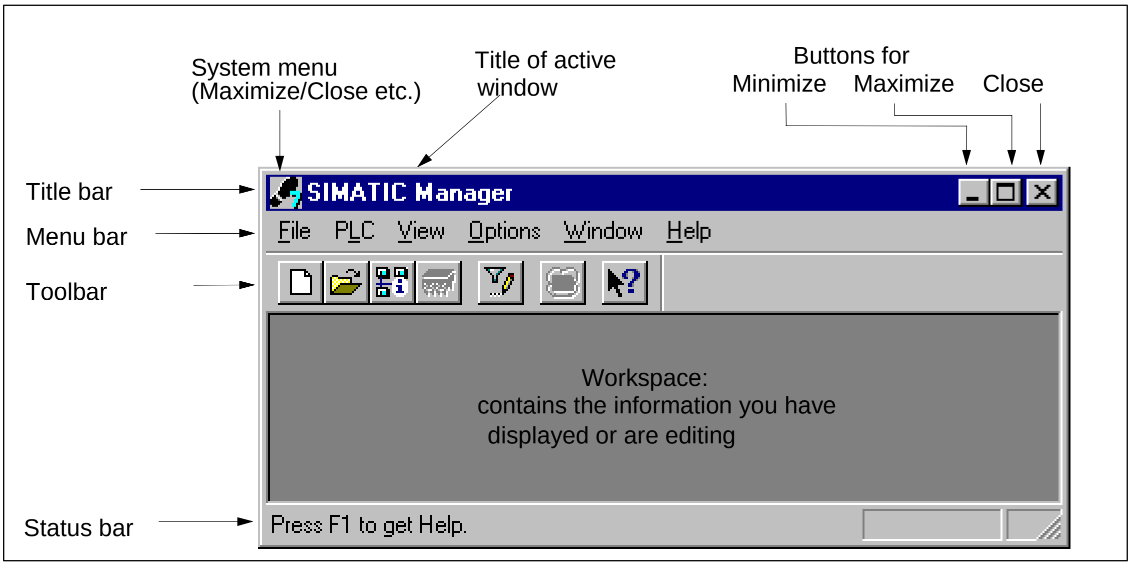
当打开一个对象时，应用程序将自动调用合适的软件组件来显示或编辑对象的内容。

继续...

下面，我们将介绍对象编辑的基本操作。请对该主题给予适当的重视，因为所有后续的主题都将基于这些基本操作。

5.5.2 窗口排列

窗口的标准组成如下图所示：



标题栏与菜单栏

标题栏与菜单栏始终位于窗口的顶部。标题栏包含窗口的标题以及对窗口进行控制的图标。菜单栏包含窗口中可供使用的所有菜单。

工具栏

工具栏包含有许多图标(或工具按钮)，这些图标提供了通过单击鼠标来执行经常使用以及当前可供使用的菜单项命令的快捷方式。当您将光标短暂放置在按钮上时，将显示对各个按钮功能的简短描述以及其它附加信息。

如果在当前组态中不能访问某个按钮，则该按钮将显示为灰色。

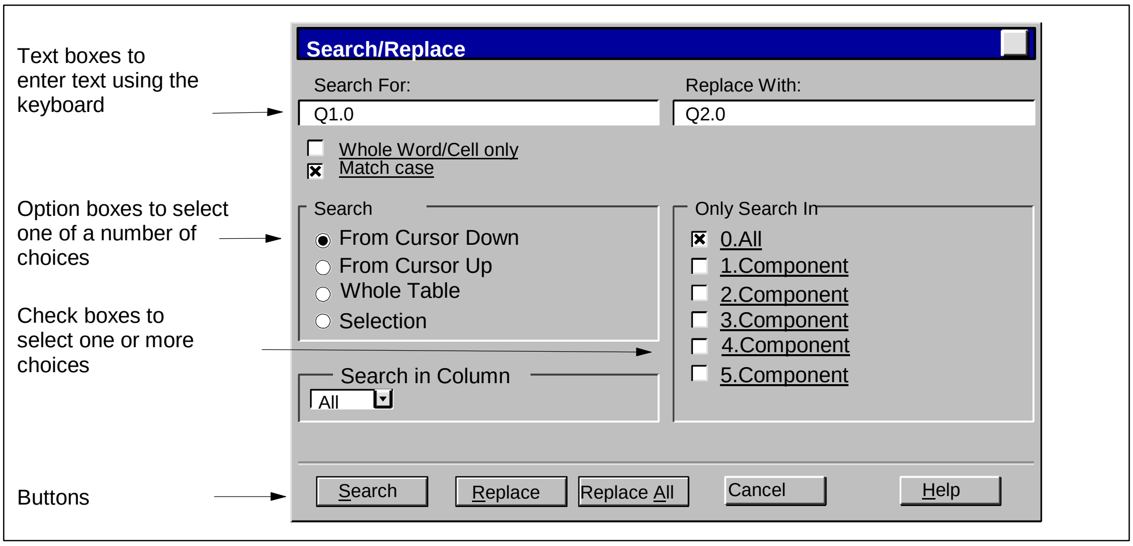
状态栏

状态栏显示了与上下文有关的信息。

5.5.3 对话框中的元素

在对话框中进行输入

在对话框中，可输入执行特定任务所需的各种信息。下图中的实例可用来说明对话框中最常见的组件。

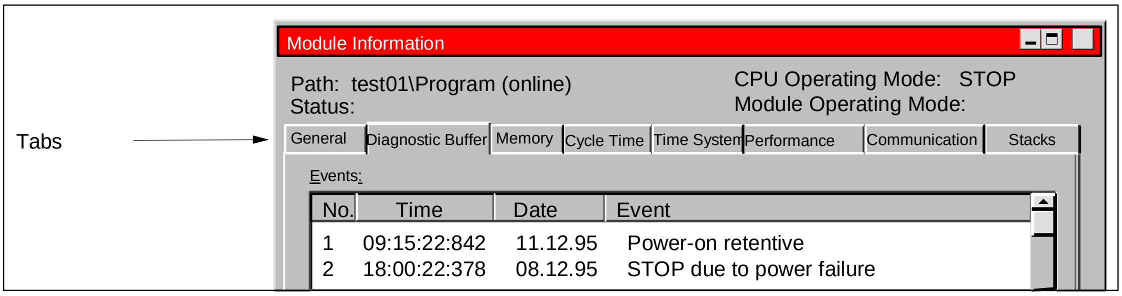


列表框与组合框

文本框旁边有时有一个方向向下的箭头。此箭头表示在该文本框中还存在许多其它选项可供选择。单击箭头可打开一个列表框或组合框。如果单击列表框中的条目，则该条目将自动显示在文本框中。

对话框中的标签

通过将对话框分为多个标签卡(见下图)，用标签来组织某些对话框的内容，可以提高信息的清晰度。



标签卡的内容沿着对话框的上边沿显示在标签上。为使某个特定的标签卡跳转到前景中，只需单击该标签即可。

5.5.4 创建和管理对象

一些基本的处理步骤与对象的类型无关，对所有的对象都相同。这里对这些标准的处理顺序作了总结。有关标准步骤的知识需要继续阅读本手册的其它章节。

处理对象时通常的步骤顺序为：

- 创建对象
- 选择对象
- 执行对象的动作(例如，复制、删除)。

设置创建新项目/库的路径

新建的用户项目、库和多项目保存在缺省的文件夹“\Siemens\Step7\S7proj”中。如果要将它们保存在其它的文件夹中，第一次保存项目、库和多项目之前，先为这些对象设置自定义的路径。为此，可选择菜单命令**选项 > 自定义**。在显示的对话框中的“常规”标签中，指定用来保存新项目或库的路径名称。

创建对象

STEP 7 向导“新建项目”可为创建新项目和插入对象提供支持。使用菜单命令**文件 >“新项目”向导**来打开该向导。在显示的对话框中，可以设置项目的结构，然后向导就会为您创建项目。

如果您不想使用向导，可以使用菜单命令**文件 > 新建**来创建项目和库。这些对象构成了对象体系的起始点。所有其它的项目假如没有自动创建，可以使用插入菜单中的命令在体系中创建。例外的是 SIMATIC 站中的模块，它们在组态硬件时创建或使用“新建项目”向导创建。

打开对象

要在详细资料中打开对象，可以有多种方法：

- 双击对象图标
- 选择对象，然后使用菜单命令编辑 > 打开对象。这仅适用于不是文件夹的对象。

当打开一个对象后，可以创建或改变它的内容。

当打开一个不包含其它对象的对象时，将以一个合适的软件组件在新窗口中显示它的内容，以供编辑。您不能改变那些内容正用于其它地方的对象。

注意

例外：对于可编程的模块(当您双击它们时)和站组态，站显示为文件夹。如果双击“硬件”对象，将会启动用于组态硬件的应用程序。选择站和选择菜单命令**编辑 > 打开对象**的效果相同。

构建对象体系

使用“新建项目”向导来创建对象体系。在打开文件夹后，它所包含的对象将会显示在屏幕上。然后，就可以使用插入菜单创建更多的对象，例如，项目的附加站。插入菜单中，只有那些能被插入到当前文件夹的对象的命令处于激活状态。

设置对象属性

对象属性是决定对象特性的对象数据。创建新对象后，将自动出现设置对象属性用的对话框，必须在其中为对象设置属性。设置的属性也可以在以后改变。

使用菜单命令**编辑 > 对象属性**，打开一个对话框，可在其中显示或设置所选对象的属性。

使用菜单命令**编辑 > 特殊对象属性**，可以打开对话框，输入操作员监控功能和组态消息所需的数据。

例如，为了显示块的特殊对象属性，以进行操作员监控，必须将块标记为与操作员监控相关，也就是要在块属性的“属性”标签中，将系统属性“s7_m_c”的值设置为“true”。

注意

不能显示或修改“系统数据”文件夹和“硬件”对象的属性。

不能在只读项目的对象属性对话框中进行写入操作。这种情况下，输入框变为灰色。

在显示可编程模块的属性时，为了保持一致性，不能对显示的参数进行编辑。要编辑参数，必须打开“组态硬件”应用程序。

剪切、粘贴、复制

大多数的对象都可以象通常在 Windows 中那样进行剪切、粘贴或复制。这些功能的菜单命令位于编辑菜单中。

也可以通过拖放来复制对象。如果试图将对象移动或复制到非法的目标地址，光标将显示一阻止符号作为警告。

在复制对象时，也将复制它以下的整个体系。这使得在一个自动化任务中创建的组件可以重复地使用。

注意

“连接”文件夹中的连接表不能复制。请注意，当复制操作员相关文本的列表时，将只接受那些目标对象中已安装的语言。

在复制对象章节中有逐步的复制操作指南。

将对象重命名

SIMATIC管理器将给一些新的对象分配标准名称。这些名称通常由对象类型(如果在同一个文件夹中可以创建多个该类型的对象)和数字构成。

例如，第一个 S7 程序被命名为“S7 Program(1)”，第二个为“S7 Program(2)”等等。而由于每个文件夹中只能有一个符号表，就简单地称其为“符号”。

大多数的对象，都可以修改名称，可用更加与内容相关的名称来命名它。

对于项目，路径的目录名不能超过 8 个字符。否则，在归档或使用“用于 M7 的 C”(Borland 编译器)时会出错。

可以直接或使用对象属性来改变对象的名称。

直接地：

- 缓慢地单击两次所选对象的名称，将在文本的周围出现一个框。然后，就可使用键盘来编辑名称。

使用菜单：

在项目窗口中选择所需的对象，然后选择菜单命令**编辑 > 重命名**。在文本的周围出现一个框。然后，就可使用键盘来编辑名称。

如果不允许您改变名称：

- 如果不允许您改变对象名，对话框的输入域将显示为灰色，其中显示当前的名称，此时不能进行文本输入。

注意

在编辑名称时，将鼠标指针移动到名称框以外，执行其它的动作(例如，选择菜单命令)，编辑过程就被终止。如果名称是允许的，将接受改变后的名称。

在重命名对象章节中有逐步的重命名操作指南。

移动对象

使用SIMATIC管理器，可以将对象从一个文件夹移动到另一个文件夹，即使目标地址位于另一个项目中。在移动文件夹时，它的内容也全部移动。

注意

不能移动下列对象：

- 连接
- 在线视图中的系统数据块(SDB)
- 在线视图中的系统功能(SFC)和系统功能块(SFB)

在移动对象章节中有逐步的移动操作指南。

排序对象

可以根据对象的属性，在详细资料中对其排序(菜单命令**视图 > 细节**。为此，单击所需属性的相应标题。再次单击，将返回到原来的排序。同一类型的块以它们的数字顺序来排序，例如，FB1，FB2，FB11，FB12，FB21，FC1。

缺省的排序顺序

在重新打开项目后，在详细资料中对象根据缺省的排序顺序来显示。实例：

- 块的显示顺序为“系统数据，OB，FB，FC，DB，UDT，VAT，SFB，SFC”。
- 在项目中，先显示所有的站，接下来显示 S7 程序。

因此在详细资料中，缺省的排序顺序不是字母数字的升序或降序。

恢复缺省排序顺序

排序后，例如，单击列标题“对象名”排序，可以按以下步骤恢复缺省顺序：

- 在详细资料中单击列标题“类型”。
- 关闭项目，然后再次打开。

删除对象

您可以删除文件夹和对象。如果删除了一个文件夹，它所包含的全部内容也将被删除。

不能撤消删除步骤。如果您不能确定是否真的不再需要一个对象，最好先为整个项目归档。

注意

不能删除下列对象：

- 连接
- 在线视图中的系统数据块(SDB)
- 在线视图中的系统功能(SFC)和系统功能块(SFB)

在删除对象章节中有逐步的删除操作指南。

5.5.5 选择对话框中的对象

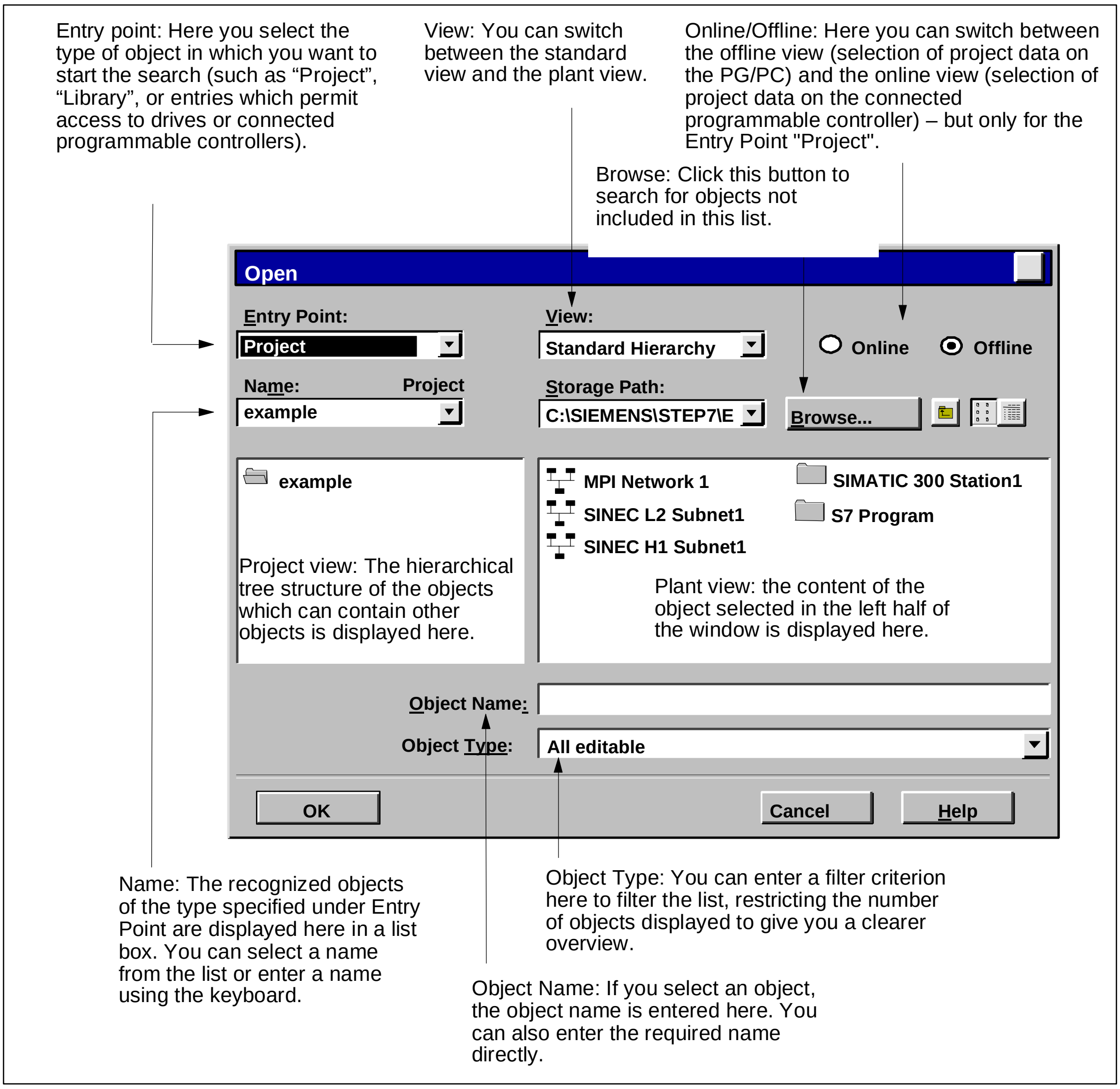
在对话框(浏览器)中选择对象是在大量不同编辑步骤中经常需要的一个动作。

调用浏览器

例如，可在硬件组态应用程序中使用诸如站 > 新建/打开等菜单命令调用浏览器对话框(基本应用程序窗口“SIMATIC 管理器”除外)。

浏览器对话框的结构

在浏览器中，存在下图所示的各种选项。



5.5.6 会话存储器

SIMATIC 管理器可保存窗口内容(即，打开的项目和库)以及窗口布局。

- 使用菜单命令**选项 > 自定义**，定义是否在一个会话结束时保存窗口内容和布局。在下一个会话开始时，恢复这些窗口内容和布局。在打开的项目中，光标位于最后选中的那个文件夹上。
- 使用菜单命令**窗口 > 保存设置**，可以保存当前窗口内容和窗口布局。
- 使用菜单命令**窗口 > 恢复设置**，可以恢复用菜单命令**窗口 > 保存设置**保存过的窗口内容和布局。在打开的项目中，光标位于最后选中的那个文件夹上。

注意

不保存在线项目的窗口内容、“可访问节点”窗口的内容以及“S7存储卡”窗口的内容。

在会话结束时，不保存任何为访问可编程控制器
(S7-300/S7-400)而输入的密码。

5.5.7 改变窗口排列

要逐个层叠所有显示窗口，可采用下列方法：

- 选择菜单命令**窗口 > 排列 > 层叠**。
- 按下组合键 SHIFT + F5。

要将所有显示窗口在屏幕上从上到下排列，可选择菜单命令**窗口 > 排列 > 水平**。

要将所有显示窗口在屏幕上从左到右排列，可选择菜单命令**窗口 > 排列 > 垂直**。

5.5.8 保存和恢复窗口布局

STEP 7应用程序有一个特点，就是使您能够保存当前的窗口布局，并在将来某个阶段对其进行恢复。可以使用菜单命令**选项 > 自定义**在“常规”标签中进行设置。

保存什么？

保存窗口布局时，将记录以下信息：

- 主窗口的位置
- 已打开的项目和库及其各自的窗口位置
- 级联窗口的次序

注意

在线项目的窗口内容、“可访问节点”窗口的内容以及“S7存储卡”窗口的内容均不能保存。

保存窗口布局

要保存当前窗口布局，请选择菜单命令**窗口 > 保存设置**。

恢复窗口布局

要恢复保存的窗口布局，选择菜单命令**窗口 > 恢复设置**。

注意

在恢复窗口时，将只详细显示包含有保存窗口时所选对象的部分结构。

5.6 键盘操作

5.6.1 键盘控制

国际通用的键名称	德国的键名称
HOME	POS1
END	ENDE
PAGE UP	BILD AUF
PAGE DOWN	BILD AB
CTRL	STRG
ENTER	Eingabetaste
DEL	ENTF
INSERT	EINFG

5.6.2 用于菜单命令的组合键

每个菜单命令都可通过键入与 ALT 键的键组合来选择。

按所显示的顺序按下下列键：

- ALT 键
- 所需菜单名中带下划线的字母(例如，ALT、对应于菜单“文件”的字母 F – 如果菜单栏中包含有菜单“文件”的话)。菜单将打开。
- 所需菜单命令中带下划线的字母(例如，对应于菜单命令“新建”的字母 N)。如果菜单命令具有子菜单，则子菜单也将打开。按上述步骤操作，直到通过键入相关字母选择了整个菜单命令。

一旦输入完毕组合键中的最后一个字母，则马上执行菜单命令。

实例：

菜单命令	键组合
文件 > 归档	ALT、F、A
窗口 > 布局 > 级联	ALT、W、A、C

用于菜单命令的快捷键

命令		快捷键
新建	(文件菜单)	CTRL+N
打开	(文件菜单)	CTRL+O
另存为	(“文件”菜单)	CTRL+S
打印 > 对象表	(“文件”菜单)	CTRL+P
打印 > 对象内容	(“文件”菜单)	CTRL+ALT+P
退出	(“文件”菜单)	ALT+F4
剪切	(“编辑”菜单)	CTRL+X
复制	(“编辑”菜单)	CTRL+C
粘贴	(“编辑”菜单)	CTRL+V
删除	(“编辑”菜单)	DEL
全选	(“编辑”菜单)	CTRL+A
重命名	(“编辑”菜单)	F2
对象属性	(“编辑”菜单)	ALT+RETURN
打开对象	(“编辑”菜单)	CTRL+ALT+O
编译	(“编辑”菜单)	CTRL+B
下载	(PLC菜单)	CTRL+L
诊断/设置 > 模块状态	(“PLC”菜单)	CTRL+D
诊断/设置 > 运行模式	(“PLC”菜单)	CTRL+I
更新	(“视图”菜单)	F5

命令		快捷键
更新在线视图中的可见CPU的状态显示		CTRL+F5
自定义	(“选项”菜单)	CTRL+ALT+E
参考数据 > 显示	(“选项”菜单)	CTRL+ALT+R
布局 > 级联	(窗口菜单)	SHIFT+F5
布局 > 水平	(窗口菜单)	SHIFT+F2
布局 > 垂直	(窗口菜单)	SHIFT+F3
上下文关联帮助	(帮助菜单)	F1 (如果当前存在上下文，例如所选的菜单命令，则相关的帮助主题将打开。否则，显示帮助目录页面。)

5.6.3 用于移动光标的组合键

在菜单条/弹出式菜单中移动光标

目的	按下
移动到菜单栏	F10
移动到弹出式菜单	SHIFT+F10
移动到包含键入时加下划线的字母或数字的菜单	ALT+菜单标题中的加下划线字符
选择其加下划线字母或数字与所键入的字母相一致的菜单命令	菜单命令中的加下划线字符
向左移动一个菜单命令	左箭头
向右移动一个菜单命令	右箭头
向上移动一个菜单命令	上箭头
向下移动一个菜单命令	下箭头
激活选中的菜单命令	ENTER
取消选择菜单名称或闭合打开的菜单，然后返回文本	ESC

编辑文本时移动光标

移动	按下
向上移动一行或在只包含一行的文本中向左移动一个字符	上箭头
向下移动一行或在只包含一行的文本中向右移动一个字符	下箭头
向右移动一个字符	右箭头
往左移动一个字符	左箭头

移动	按下
向右移动一个字	CTRL+右箭头
向左移动一个字	CTRL+左箭头
移动到行头	HOME
移动到行末	END
移动到前一个画面	PAGE UP
移动到下一个画面	PAGE DOWN
移动到文本头	CTRL+HOME
移动到文本末	CTRL+END

编辑表格时移动光标

移动	按下
向上一行	上箭头
向下一行	下箭头
向左移动一个字符或单元	右箭头
向右移动一个字符或单元	左箭头
移动到行头	CTRL+右箭头
移动到行末	CTRL+左箭头
移动到单元头	HOME
到单元末	END
移动到前一个画面	PAGE UP
移动到下一个画面	PAGE DOWN
移动到表头	CTRL+HOME
移动到表末	CTRL+END
只在符号表中：移动到“符号”列	SHIFT+HOME
只在符号表中：移动到“备注”列	SHIFT+END

在对话框中移动光标

目的	按下
从一个输入框移动到下一个输入框(从左到右、从上到下)	TAB
反向移动一个输入框	SHIFT+TAB
移动到包含所键入加下划线字母或数字的输入框或选项	ALT+菜单标题中的加下划线字符
在选项列表中选择	箭头键
打开选项列表	ALT+下箭头
选择或取消选择列表中的一个条目	空格键
确认输入，关闭对话框(“确定”按钮)	ENTER
不保存改变，关闭对话框(“取消”按钮)	ESC

5.6.4 用于选择文本的组合键

选择或取消选择文本	按下
往右每次移动一个字符	SHIFT+向右箭头键
往左移动一个字符	SHIFT+向左箭头键
移到命令行的起始处	SHIFT+HOME
移到命令行的末尾	SHIFT+END
在表格中移动一列	SHIFT+SPACE
往上移动文本中的一行	SHIFT+向上箭头键
往下移动文本中的一行	SHIFT+向下箭头键
移到前一屏	SHIFT+PAGE UP
移到下一屏	SHIFT+PAGE DOWN
移到文件开始处的文本	CTRL+SHIFT+HOME
移到文件结束处的文本	CTRL+SHIFT+END

5.6.5 用于访问在线帮助的组合键

目的	按下
打开帮助	F1 (如果当前存在有上下文，例如所选的菜单命令，则相关的帮助主题将打开。否则，显示帮助目录页面。)
激活用于上下文关联帮助的问号标记符号	SHIFT+F1
关闭帮助窗口并返回到应用程序	ALT+F4

5.6.6 用于切换窗口的组合键

目的	按下
在窗格之间切换	F6
返回到前一个窗格(如果没有任何可处理的窗口)	Shift+F6
在文档窗口和文档中的可处理窗口(例如，变量声明窗口)之间切换。 如果没有任何可处理的窗口，则可使用该组合键返回到前一个窗格。	Shift+F6
文档窗口切换	Ctrl+F6

目的	按下
返回到前一个文档窗口	Shift+Ctrl+F6
在非文档窗口之间切换(应用程序框架与应用程序框架中的可处理窗口； 当返回到框架时，该组合键将激活上一次曾激活的文档窗口)	Alt+F6
返回到前一个非文档窗口	Shift+Alt+F6
关闭活动窗口	Ctrl+F4

6 建立和编辑项目

6.1 项目结构

项目用于存储您在提出自动化解决方案时所创建的数据和程序。项目所汇集的数据包括：

- 关于模块硬件结构及模块参数的组态数据；
- 用于网络通讯的组态数据，以及
- 用于可编程模块的程序。

在创建项目时的主要任务就是准备这些数据，以备编程使用。

数据将以对象的形式存储在项目中。对象在项目中按树形结构排列(项目体系)。项目体系在项目窗口中的显示类似于Windows资源管理器中的显示。只是对象图标的外观不同。

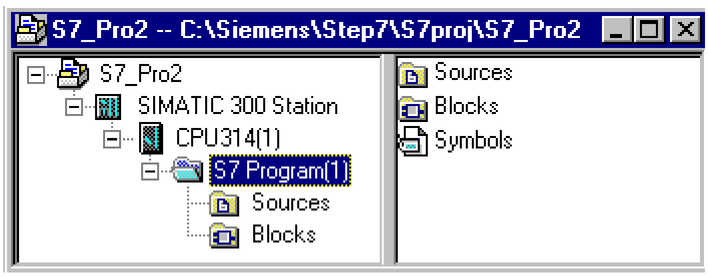
项目体系顶端的结构如下：

1. 第一层： 项目
2. 第二层： 子网、站、或S7/M7程序
3. 第三层： 取决于第二层的对象。

项目窗口

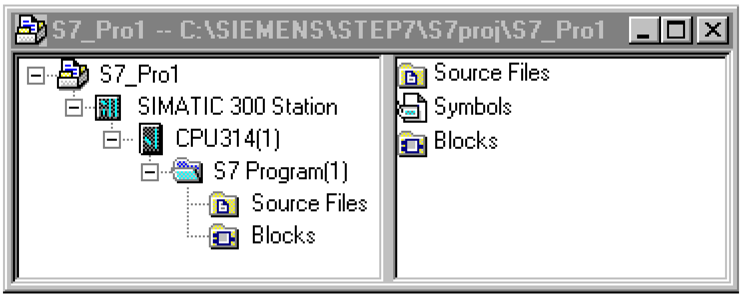
项目窗口分为两半部分：左半部分表示项目的树形结构。右半部分表示所选视图左半部分已打开的对象所包含的对象(大图标、小图标、列表或详细信息)。

单击窗口左半部分中含有加号的方框即可显示项目的完整树形结构。所生成的结构在某种程度上类似于下图。



对象体系的最上端是代表整个项目的对象“S7_Pro1”的图标。它可用于显示项目属性，并可用作网络文件夹(用于对网络进行组态)、站文件夹(用于对硬件进行组态)、以及S7或M7程序的文件夹(用于创建软件)。项目中的对象在选择项目图标时均将显示在项目窗口的右半部分。该类型对象体系最上端的对象(库以及项目)构成了用于对对象进行选择的对话框的起始点。

项目视图



在项目视图中，既可在组件视图“离线”中显示可编程设备可用数据的项目结构，也可在组件视图“在线”中显示可编程控制系统上可用数据的项目结构。

如果安装了相应的选项包，还可以设置一个附加的视图：工厂视图：

注意

只能在“离线”视图中对硬件和网络进行组态。

6.2 设置过项目

6.2.1 创建项目

要使用项目管理框架构造自动化任务的解决方案，需要创建一个新的项目。新项目的创建目录是您通过菜单命令**选项 > 自定义**在“常规”标签中为项目设定的目录。

注意

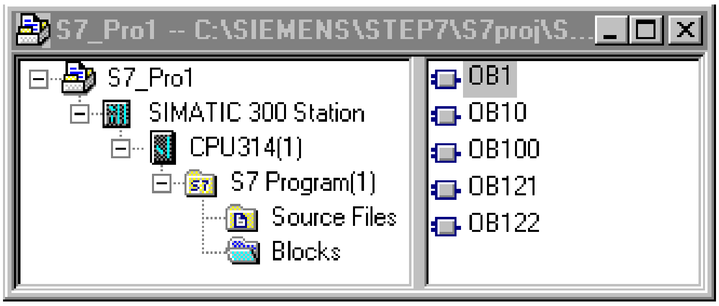
SIMATIC管理器允许使用长度多于8个字符的名称。不过，项目目录的名称裁减到8个字符。因此，项目名称的前8个字符必须有所不同。名称不区分大小写。

可在手动创建项目或使用向导创建项目。

使用向导创建项目

创建新项目的最简单方法就是使用“新项目”向导。使用菜单命令**文件 >** “**新项目**”**向导**来打开该向导。向导提示在对话框中输入所要求的详细资料，然后创建项目。除了站、CPU、程序文件夹、源文件夹、块文件夹以及OB1之外，还可以选择已存在的OB1，进行错误和报警处理。

下图显示了通过向导创建一个新项目的实例。



手动创建项目

可在SIMATIC管理器中使用菜单命令**文件 > 新建**来创建一个新项目。它已经包含“MPI子网”对象。

其它步骤

编辑项目时，可自由选择大多数任务的执行顺序。一旦创建了项目，可以选择以下方法之一：

- 首先组态硬件，然后为其创建软件，或
- 通过创建独立于所有已组态硬件的软件来启动。

方法1：首先组态硬件

如果希望首先组态硬件，那么可按“通过 STEP 7 组态硬件手册”第 2 卷所述执行操作。完成该操作时，创建软件所要求的“S7 程序”和“M7 程序”文件夹已经插入。然后继续插入创建程序所需要的对象。之后创建可编程模块的软件。

方法2：首先创建软件

还可以不必首先组态硬件就创建软件；可在以后组态硬件。不必为了输入程序而设置站的硬件结构。

基本步骤如下：

1. 在项目中插入所要求的软件文件夹(不带站或CPU的S7/M7程序)。
   在此，可简单确定程序文件夹是否包含S7硬件或M7硬件。
2. 之后创建可编程模块的软件。
3. 组态硬件。
4. 一旦组态好硬件，可以将M7或S7程序链接到CPU中。

6.2.2 插入站

在项目中，站代表了可编程控制器的硬件结构，并包含有用于组态和给各个模块进行参数分配的数据。

使用“新建项目”向导创建的新项目已经包含有一个站。否则，可以使用菜单命令**插入 > 站**来创建新站。

可选择一个下列站点：

- SIMATIC 300 站
- SIMATIC 400 站
- SIMATIC H 站
- SIMATIC PC 站
- PC/可编程设备
- SIMATIC S5
- 其它站，即非 SIMATIC S7/M7 和 SIMATIC S5 的站

可使用预先设置的名称插入站 (例如，SIMATIC 300 站(1)、SIMATIC 300 站(2)等)。如果愿意，也可以用相关的名称替换站的名称。

关于逐步插入工作站的向导介绍，请参见插入工作站。

组态硬件

当组态硬件时，您可指定 CPU，并可借助于模块目录，指定可编程控制器中的所有模块。双击站点，即可启动硬件配置应用程序。

对于在组态中创建的每个可编程模块，一旦保存完毕并退出硬件配置，将自动创建一个S7或M7程序以及连接表(“连接”对象)。使用“新建项目”向导创建的项目已经包含有这些对象。

关于逐步组态的向导介绍，请参见组态硬件。更多详细信息，请参见对站进行组态的基本步骤。

创建连接表

将为每个可编程模块自动创建一个(空白)连接表(“连接”对象)。连接表用于定义网络中的可编程模块之间的通讯连接。打开时，将显示一个包含有表格的窗口，可在该表格中定义可编程模块之间的连接。

有关详细信息，请参见在项目内连网站。

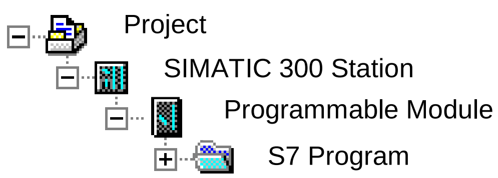
下一步

创建硬件配置后，可以编写可编程模块的软件(另请参见插入 S7/M7 程序)。

6.2.3 插入S7/M7程序

用于可编程模块的软件存储在对象文件夹中。对于 SIMATIC S7 模块，该对象文件夹被称为“S7 程序”，对于 SIMATIC M7，该对象文件夹被称为“M7 程序”。

下图所示为 SIMATIC 300 站的可编程模块中的一个 S7 程序实例。



已存在的组件

每个可编程模块都有一个自动创建的 S7/M7 程序，用作软件容器：

下列对象已经位于新创建的 S7 程序中：

- 符号表(“符号”对象)
- 包含第一个块的“块”文件夹
- 用于源文件的“源文件”文件夹

下列对象已经位于新创建的 M7 程序中：

- 符号表(“符号”对象)
- “块”文件夹

创建S7块

希望创建语句表、梯形图或功能块图程序。为此，选择已存在的“块”对象，然后选择菜单命令**插入 > S7块**。在子菜单中，可选择要创建的块类型(例如数据块、用户自定义的数据类型(UDT)、功能、功能块、组织块或变量表)。

现在可打开(空)块，开始输入语句表、梯形图或功能块图程序。可在创建逻辑块时的基本过程以及“语句表、梯形图和功能块图”手册中获得关于本操作的更多详细信息。

注意

可以位于用户程序中的对象“系统数据”(SDB)是由系统创建的。可以打开该对象，但由于一致性原因，不能对其进行改变。在已装载程序后，它可用于改变组态，并将所作的改变下载至可编程控制器。

使用来自标准库的块

还可以使用与软件同时到货的标准库中的块来创建用户程序。使用菜单命令**文件 > 打开**来访问库。可在使用库进行工作以及在线帮助中获得关于使用标准库以及创建个人库的更多信息。

创建源文件/CFC图表

希望以特定的编程语言或CFC图表来创建源文件。为此，在S7程序中选择“源文件”对象或“图表”对象，然后选择菜单命令**插入 > S7软件**。在子菜单中，可以选择与编程语言相匹配的源文件。现在可打开空的源文件，开始输入程序。可在关于在STL源文件中编程的基本信息。.

创建M7程序

希望为M7中的可编程控制器的操作系统RMOS创建程序。为此，选择M7程序，然后选择菜单命令**插入 > M7软件**。在子菜单中，可以选择与编程语言或操作系统相匹配的对象。现在可打开所创建的对象，访问相关编程环境。

创建符号表

创建 S7/M7 程序时，会自动创建一个(空)的符号表(“符号”对象)。打开符号表时，“符号编辑器”窗口会打开，显示一个符号表，可在其中定义符号。可在在符号表中输入多个共享符号。

插入外部源文件

可通过任何 ASCII 编辑器来创建和编辑源文件。然后可将这些文件导入到项目中、编译，以便创建单个块。

编译导入的源文件时所创建的块保存在“块”文件夹中。

可在插入外部源文件。

6.2.4 编辑项目

打开项目

要打开现有项目，请使用菜单命令**文件 > 打开**。然后在紧接着出现的对话框中选择一个项目。于是项目窗口打开。

注意

如果您需要的项目没有显示在项目列表中，则单击“浏览”按钮。然后在浏览器中，可以搜索其它的项目(包括在项目列表中所找到的所有项目)。可以使用菜单命令**文件 > 管理**更改项目列表中的条目。

复制项目

可使用菜单命令**文件 > 另存为**，通过用另一个名称保存项目来复制项目。可使用菜单命令**编辑 > 复制**来复制部分项目，如站、程序、块等。有关逐步复制项目的介绍，请参见复制项目和复制部分项目。

删除项目

可使用菜单命令**文件 > 删除**来删除项目。可使用菜单命令**编辑 > 删除**来删除部分项目，如站、程序、块等。有关逐步删除项目的介绍，请参见删除项目和删除部分项目。

6.2.5 检查所用软件包的项目

如果您正在编辑的项目包含了使用另一个软件包创建的项目，那么编辑该项目时需使用该软件包。

无论您使用什么编程设备来操作多项目、项目或库，STEP 7 都会显示完成该操作所需要的软件包及其版本，为您提供帮助。

在下列条件下，所需软件包的这种信息是完整的：

- 如果项目(或多项目中的所有项目)或库是使用 V5.2 以上的 STEP 7 创建的。
- 如果您自己已经检查了创建项目时所使用的软件包。为此，首先要转到 SIMATIC 管理器，并选择相关的项目。然后选择菜单命令**编辑 > 对象属性**。在所显示的对话框中，选择“所需要的软件包”标签。该标签中的信息将告诉您是否应检查项目的软件包。

6.3 管理多语言文本

6.3.1 管理多语言文本

STEP 7 有可能做到：导出在某个项目中以一种语言创建的文本、翻译该文本、重新导入文本、并以译文显示文本。

下列文本类型可以用一种以上语言管理：

- 标题和注释
  - 块标题和块注释
  - 网络标题和网络注释
  - 来自 STL 程序的行注释
  - 来自符号表、变量声明表、用户自定义数据类型和数据块的注释
  - HiGraph 程序中的注释、状态名称和转换名称
  - S7-Graph 程序中的步骤名称和步骤注释的扩展
- 显示文本
  - 由 STEP 7、S7-Graph、S7-HiGraph,或 S7-PDIAG 生成的消息文本
  - 系统文本库
  - 特定用户文本库
  - 操作者相关的文本
  - 用户文本

导出

导出所选择的对象下的所有块和符号表。为每个文本类型创建导出文件。文件包含源语言列和目标语言列。源语言文本不得改变。

导入

为选定对象下面的所有块和符号表执行导入。在导入期间，目标语言列(右边的列)的内容会集成到所选择的对象中。只接受与源语言列中的现有文本相匹配的文本。

改变语言

当改变语言时，可以从向所选择的项目导入期间指定的所有语言中选择。“标题和注释”的语言改变只适用于所选择的对象。“显示文本”的语言改变总是适用于整个项目。

删除语言

当语言删除时，所有使用这种语言的文本都从内部数据库中删除。

项目中应始终有一种语言可用作参考语言。例如，可以是本地语言。这种语言不应删除。在导出和导入期间，请始终指定该参考语言作为源语言。目标语言可以根据要求设置。

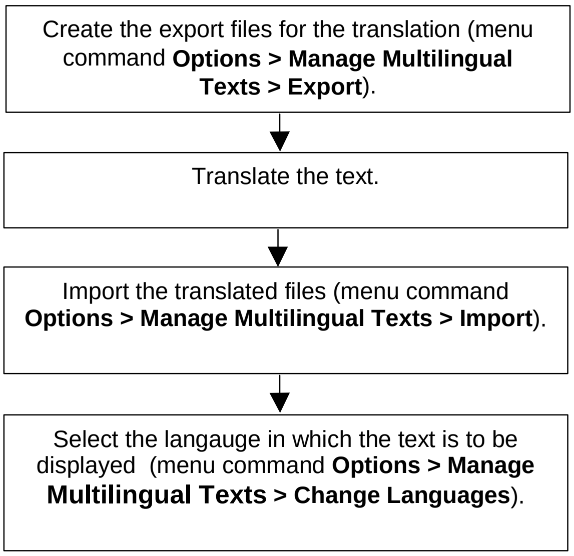
重新组织

在重新组织期间，语言会改变为当前设置的语言。当前设置语言是选作“未来块的语言”的语言。重新组织只影响标题和注释。

注释管理

可以规定在以多语言管理文本的项目中如何管理块的注释。

基本步骤



6.3.2 多语言文本的类型

为了进行导出，将为每种类型的文本创建一个单独的文件。该文件将以文本类型作为其名称，已导出格式作为其扩展名(文本类型.格式：例如，SymbolComment.CSV 或 SymbolComment.XLS)。不符合命名规范的文件将不能用作源文件和目标文件。

项目中可翻译的文本可分为下列文本类型：

文本类型	描述
BlockTitle	块标题
BlockComment	块注释
NetworkTitle	网络标题
NetworkComment	网络注释
LineComment	STL中的行注释
InterfaceComment	Var_Section注释(代码块中的声明表)以及 UDT注释(用户自定义的数据类型)以及 数据块注释
SymbolComment	符号注释
S7UserTexts	由用户输入的可在显示设备上输出的文本
S7SystemTextLibrary	集成到消息中的系统库的文本在运行期间可动态更新，并可显示在PG或其它显示设备上
S7UserTextLibrary	集成到消息中的用户库的文本在运行期间可动态更新，并可显示在PG或其它显示设备上
 HiGraphStateName HiGraphStateComment HiGraphTansitionName HiGraphTransitionComment	S7-HiGraph 语句名称 语句注释 翻译名称 翻译注释
 S7GraphStateName S7GraphStateComment	S7-Graph 步骤名扩展 步骤注释

其它选项包中的编辑器(例如 ProTool、WinCC 等等)可能具有面向其它应用程序的文本类型，在此不进行描述。

6.3.3 导出文件的结构

导出文件具有如下的结构：

实例：

$_Languages		
9(1) English (USA)	7(1) German (Germany)	
$_Type(NetworkTitle)		
First character sequence to be translated	Translation	test\S7 Program(1)\Blocks\OB1
Second character sequence to be translated	Translation	test\S7 Program(1)\Blocks\OB1
Character sequence that is not to be displayed in the translation	$_hide	test\S7 Program(1)\Blocks\OB1

Souce Language　　Target Language　　Location

原则上，下列均适用：

1. 下列内容均不必进行修改、覆盖或删除：
   - 以“$_”开始的域 (这些都是关键字)
   - 语言的编号(在上述实例中：9 (1)代表源语言英语(美国)，7 (1)代表目标语言德语)。
2. 每个文件均只具有一种单个测试类型的文本。在实例中，文本类型为NetworkTitle ($_Type(NetworkTitle)。在导出文件本身的介绍性文本中包含对该文件进行编辑的翻译员的守则。
3. 与文本或注释有关的其它信息必须始终出现在类型定义($_Type…) 的前面或最后一行的后面。

注意

如果目标语言的行已经被“512(32) $_Undefined”所覆盖，则当文件导出时，将不指定任何目标语言。为进行更好概括，您可使用目标语言，例如“9(1)英语(美国)”，来替换该文本。当导入所翻译的文本时，您必须验证所推荐的目标语言，并在必要时，选择正确的语言。

您可通过输入关键字$_hide来隐藏不以目标语言显示的文本。这不适用于关于变量的注释(InterfaceComment)，也不适用于符号(SymbolComment)。

导出文件格式

可指定将以何种格式保存导出文件。

如果您已经决定使用CSV格式，那么，在使用Excel进行编辑时，您必须注意，只有使用“打开”对话框，才能在Excel中正常打开CSV文件。**通过在资源管理器中进行双击操作来打开CSV文件将经常导致打开的文件无法使用**。如果您使用下列步骤，您将会发现在Excel中使用CSV文件进行工作将更容易：

1. 在Excel中打开导出文件。
2. 将文件另存为XLS文件。
3. 翻译XLS文件的文本。
4. 在Excel中将XLS文件另存为CSV格式的文件。

注意

不必重新命名导出文件。

6.3.4 管理未安装其语言字体的用户文本

您可导出未在操作系统中安装其语言字体的用户文本，对其进行翻译，然后将其重新导入，并进行保存，以供您的项目使用。

然而，这样的文本将只能显示在已经安装了相应语言字体的计算机上。

例如，如果您必须将用户文本翻译为俄文，但在操作系统中没有安装 Cyrillic 字体，那么，您可如下进行操作：

1. 导出源语言为“英语”、目标语言为“俄语”的要翻译的用户文本。
2. 将所导出的文件发送给翻译员，该翻译员一定要有Cyrillic字体。
3. 导入所翻译的导出文件。
   结果：现在，在有英语和俄语的计算机上都可以使用该项目。
4. 保存整个项目，并发送给将要使用俄语文本的客户，客户因此要有Cyrillic字体，以便显示。

6.3.5 优化翻译源文本

可通过组合不同的术语和表达式来准备用于翻译的源材料。

实例

在准备之前(导出文件)：

$_Languages		
9(1) English (USA)	9(1) English (USA)	
$_Type(SymbolComment)		
Auto-enab.		
Automatic enable		
Auto-enable		

Source Language　　Target Language

组合成一个单个表达式：

$_Languages		
9(1) English (USA)	9(1) English (USA)	
$_Type(SymbolComment)		
Auto-enab.	Auto-enable	
Automatic enable	Auto-enable	
Auto-enable	Auto-enable	

Source Language　　Target Language

在准备之后(也就是说，在导入以及随后的导出之后)：

$_Languages		
9(1) English (USA)	9(1) English (USA)	
$_Type(SymbolComment)		
Auto-enable	Auto-enable	

Source Language　　Target Language

6.3.6 优化翻译过程

如果您拥有的项目，其结构和文本都类似于前一个项目，则您可对翻译过程进行优化。

特别对于通过复制，然后对其进行修改所创建的文件，建议按下列步骤操作。

前提条件

必须有现存的已翻译的导出目标。

步骤

1. 将导出文件复制到用于保存将要翻译的新项目的项目文件夹中。
2. 打开新项目，并导出文本(菜单命令**选项 > 管理多语言文本 > 导出**)。因为导出目标已经存在，所以将询问是扩展导出目标，还是将其覆盖。
3. 单击“添加”按钮。
4. 对导出文件进行翻译(仅需要翻译新的文本)。
5. 然后导入所翻译的文本。

6.4 微型存储器卡(MMC)作为数据载体

6.4.1 关于微型存储器卡(MMC)须知

微存储卡(MMC)是插入式存储卡，例如，用于CPU 31xC或IM 151/CPU (ET 200S)。它们最显著的特征是高度压缩的设计。

新存储器概念已为 MMC 采用。下面简要描述。

MMC的内容

MMC 可作为装载存储器和数据存储设备(数据载体)。

MMC作为装载存储器

MMC包含与MMC兼容的某个CPU的全部**装载存储器**。装载存储器包含具有块(OB、DB、FC...)以及硬件组态的程序。装载存储器的内容影响CPU的功能。MMC作为装载存储器使用时，可以从它传送具有装载功能的块和硬件组态(即**下载**到CPU)。下载到CPU的块立即生效；而硬件组态只有在CPU重启动后才生效。

存储器复位的反应

在存储器复位后，存储在 MMC 上的块仍保留。

装载和删除

可以重写 MMC 上的块。

可以删除 MMC 上的块。

不能恢复已重写或删除的块。

访问MMC上的数据块

在MMC上，可以使用数据块和数据块内容，处理较大数量的数据或在用户程序中几乎不需要的数据。为此可使用新的系统操作：

- SFC 82：在装载存储器中创建数据块
- SFC 83：读装载存储器中的数据块
- SFC84：写装载存储器中的数据块

MMC和密码保护

如果安装了微存储卡(MMC)的 CPU (即在 300-C 系列的 CPU)受密码保护，那么，在 SIMATIC 管理器中(在编程设备/PC 上)打开 MMC 时，也会提示用户输入此密码。

在STEP 7中显示存储器分配

在模块状态对话框(“存储器”标签)中显示装载存储器分配时，EPROM 和 RAM 区域均会显示。

MMC 上的块显示 100% EPROM 的性能。

6.4.2 使用微型存储器卡作为数据载体

STEP 7 使用 SIMATIC 微存储器卡(MMC)的方式与使用任何其它类型外部数据存储介质的方式相同。

在确定 MMC 具有足够的容量可容纳所有要存储的数据之后，您可将操作系统的文件资源管理器中可见的任何数据传送给 MMC。.

采取这种方式，您可使其他人员也可利用属于设备的附加插图、维修说明以及功能描述。

6.4.3 存储器卡文件

生成的存储器卡文件(*.wld)将用于：

- 软件 PLC WinLC (WinAC Basis 和 WinAC RTX)以及
- SlotPLC CPU 41x-2 PCI (WinAC Slot 412 和 WinAC Slot 416)。

WinLC 或 CPU 41x-2 PCI 的块和系统数据均可同在 S7-存储器卡中一样保存在存储器卡文件中。这些文件的内容随后将与 S7-CPU 的相应存储器卡的内容保持一致。

对应于使用 STEP 7 的用户程序的下载，该文件可通过 WinLC 或 CPU 41x-2 PCI 操作面板的菜单命令下载到它们的存储器中。

就 CPU 41x-2 PCI 而言，如果 CPU 41x-2 PCI 没有进行缓冲，且只使用 RAM 卡进行操作(“自动装载”功能)，那么，当 PC 操作系统启动时，该文件将可自动进行下载。

对于 Windows 来说，存储器卡文件是“正常的”文件，借助于资源管理器，可对其进行移动、删除或使用数据介质进行传输。

更多信息，请参见 WinAC 产品的相应文档。

6.4.4 在微型存储器卡(MMC)上存储项目数据

使用STEP 7，您可将STEP 7项目的数据以及任何其它种类的数据(例如Word或Excel文件)存储在适当的CPU或编程设备(PG)/PC中的SIMATIC微型存储器卡(MMC)上。这将使您能够使用没有在其上保存项目的编程设备访问项目数据。

要求

只有将其插入到适当的 CPU 或编程设备(PG)/PC 的插槽中，并已建立一个在线连接后，才能将项目数据存储在 MMC 上。

确保 MMC 具有足够的容量，能够容纳所有要存储的数据。

可存储在MMC上的数据

在确定MMC具有足够的容量可容纳所有要存储的数据之后，您可将操作系统的文件资源管理器中可见的所有数据传送给MMC。它们包括下面的数据：

- STEP 7 的完整项目数据
- 站组态
- 符号表
- 块和源
- 以多种语言管理的文本
- 任何其它种类的数据，例如 Word 或 Excel 文件

如何保存整个项目

1. 选择**文件 > 存储卡文件**菜单命令。
2. 在“文件名”域中，输入一个指定的文件名，而不输入文件类型扩展名。
3. 在“文件类型”下拉列表中，选择文件类型“项目(*.wld)”。
4. 点击“保存”按钮。

7 用不同版本的STEP 7编辑项目

7.1 编辑版本2项目和库

STEP 7的版本V5.2不再支持V2项目中的改变。当编辑V2项目或库时，可以发生不一致性，以致于V2项目或库不再能用更旧的STEP 7版本编辑。

为了继续编辑 V2 项目或库，必须使用 STEP 7 V5.1 以前的版本。

7.2 扩展用STEP 7的从前版本创建的DP从站

可通过导入新的*.GSD文件形成的组群

如果在硬件目录中安装新的设备数据库文件(*.GSD文件)，HW Config可以接受新的DP从站。安装后，它们位于“其它域设备”文件夹中。

如果存在下列所有条件，则不能用通常方式重新组态或扩展模块化DP从站：

- 从站用 STEP 7 的从前版本组态。
- 从站在硬件目录中以类型文件而不是以*.GSD 文件出现。
- 从站上已经安装了新的*.GSD 文件。

纠正方法

如果希望使用在*.GSD文件中描述的具有**新模块**的DP从站：

- 删除 DP 从站，并再次组态。然后，DP 从站完全由*.GSD 文件、而不是由类型文件描述。

如果**不希望使用**仅在*.GSD文件中描述的**任何新模块**：

- 在硬件目录窗口中的 PROFIBUS-DP 下，选择“其它现场设备/兼容的 PROFIBUS-DP 从站”文件夹。当“旧的”类型文件由新的*.GSD 文件代替时，STEP 7 将“旧的”类型文件移动到此文件夹中。在此文件夹中，将发现可以用来扩展已组态 DP 从站的模块。

用STEP 7 V5.1 Service Pack 4中的GSD文件代替类型文件后的组群

从 STEP 7 V5.1Service Pack 4 起，类型文件要么更新，要么大量地由 GSD 文件代替。此代替只影响随 STEP 7 提供的目录配置文件，而不影响用户自行创建的目录配置文件。

其属性以前由类型文件确定、而现在由 GSD 文件确定的 DP 从站，仍位于硬件目录中的相同位置。

“旧的”类型文件不会删除，而是转移到硬件目录中的另一个位置。它们现在位于“**其它域设备\兼容的 PROFIBUS DP 从站\...**”目录文件夹中。

从V5.1 Service Pack 4起，用STEP 7扩展现有的DP组态

如果编辑用STEP 7的从前版本(早于V5.1，SP4)创建的项目，并且希望扩展模块化DP从站，那么不能使用从硬件目录的通常位置取得的模块或子模块。在这种情况下，可使用在“**其它域设备\兼容的PROFIBUS DP从站\...**”处找到的DP从站。

用STEP 7 V5.1，SP4的早先版本编辑DP组态

如果用 STEP 7 V5.1，Service Pack 4 以上版本组态“更新的”DP 从站，再用 STEP 7 的从前版本(早于 STEP 7 V5.1，SP4)编辑项目，将不能编辑此 DP 从站，因为从前版本不能识别所使用的 GSD 文件。

纠正方法：可以在 STEP 7 的从前版本中安装所需要的 GSD 文件。在此情况下，GSD 文件存储在项目中。如果随后用当前的 STEP 7 版本编辑项目，会用新安装的 GSD 文件进行组态。

7.3 用STEP 7的从前版本编辑当前组态

组态直接数据交换(横向通讯)

组态无DP主站系统的DP主站的直接数据交换：

- 不能用于 STEP7 V5.0，Service Pack 2 (或更旧版本)
- 可用于 STEP7 V5.0 版本，Service Pack 3 以上和 STEP 7 V5.1 版本以上

如果保存无DP主站系统的DP主站及其直接数据交换的已组态分配，并且继续用较旧版本的STEP 7 V5 (STEP 7 V5.0，Service Pack 2 (或更旧))编辑此项目，可能产生下列结果：

- 显示 DP 主站系统及用作 STEP 7 内部数据存储区域的从站，存放直接数据交换分配。这些 DP 从站不属于所显示的 DP 主站系统。
- 不能将新的或孤立的 DP 主站系统连接到此 DP 主站。

通过PROFIBUS-DP接口在线连接到CPU

组态无DP主站系统的PROFUBUS-DP接口：

- STEP7 V5.0，Service Pack 2 (或更旧版本)：不可能通过本接口连接到 CPU。
- 从 STEP7 V5.0，Service Pack 3 起：在编译期间，生成用于 PROFIBUS-DP 接口的系统数据；下载后，可通过此接口连接到 CPU。

7.4 从前版本的附加SIMATIC PC组态

STEP 7 V5.1项目的PC组态(直到SP 1)

从STEP 7 V5.1，Service Pack 2起，就像下载到S7-300或S7-400站一样，可以下载通讯到PC站(无需通过组态文件绕道)。然而，始终会在存储或编译操作期间生成组态文件，以便能用该办法将组态传送到目标PC站。

这样做的结果是，“更旧”的 PC 站不能解释新生成的组态文件中所包含的某些信息。STEP 7 会自动适应这种情况：

- 从 V5.1，Service Pack 2 起，如果用 STEP 7 创建新 SIMATIC PC 站组态，STEP 7 会假定目标 PC 站是在 2001 年 7 月以后的 SIMATIC NET CD 的帮助下组态的，即假设已经安装了 S7RTM (运行系统管理器)。这样生成的组态文件可以被“新”PC 站解释。
- 如果附加以前版本的 SIMATIC PC 站 组态(例如，PC 站用 STEP 7 V5.1，Service Pack 1 组态)，STEP 7 **不会**假定目标 PC 站是在 2001 年 7 月以后的 SIMATIC NET CD 的帮助下组态的。用这种方式生成的组态文件可以被“旧”PC 站解释。

如果此缺省性能不符合您的要求，可以如下描述进行修改：

在关联菜单“组态硬件”中设置：

1. 打开PC站硬件组态
2. 右击站窗口(白色区域)
3. 选择“站属性”关联菜单
4. 选中或清除“兼容性”复选框。

在“组态网络”关联菜单中设置：

1. 打开网络组态
2. 高亮显示PC站
3. 选择菜单命令**编辑 > 对象属性**
4. 在对话框中，选择“组态”标签
5. 选中或清除“兼容性”复选框。

STEP 7 V5.0项目的PC组态

从 V5.0，Service Pack 3 起，如果希望用 STEP 7 编辑 SIMATIC PC 站组态，以便组态仅由 Service Pack 3 或更高版本支持的新组件，则必须转换站：

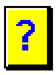
1. 在SIMATIC管理器中，高亮显示SIMATIC PC站，并选择菜单命令**编辑 > 对象属性**。
2. 在属性对话框的“功能”标签中，点击“扩展”按钮。
   然后，SIMATIC PC站被转换。现在，它只能用STEP 7 V5.0，Service Pack 3或更新版本编辑。

7.5 显示已使用最新STEP 7版本或选项包组态的模块

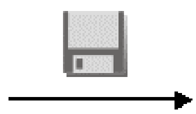
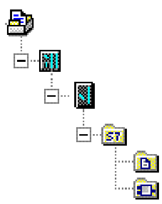
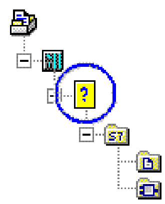
从 STEP 7 V5.1 Service Pack 3 起，所有模块均将显示，即使是使用新的 STEP 7 版本进行了组态而在“旧的”STEP 7 下无法识别的那些模块。使用选项包组态的模块也将显示，即使编程设备(PG)上没有安装用于打开给定项目的相应选项包。

在先前的STEP 7版本中，将不显示这样的模块和它们的附属对象。在当前的版本中，这些对象均是可见的，并可进行某种程度的编辑。例如，您也可使用该功能修改用户程序，即使项目是在另一个运行更新版本STEP 7的计算机上创建的，而模块(例如CPU)由于具有新的属性和新的参数，不能使用现有的更早版本的STEP 7进行组态。

STEP 7“未知的”模块将显示为一个通用的、具有下列图标的代表模块：



如果使用适当的STEP 7版本或兼容的选项包打开项目，则所有的模块都将以其标准方式显示，并可不受任何限制地编辑。

具有最新STEP 7/选项包的PG		**具有较早STEP 7/不带选项包的PG**
		
	>>>---项目数据--->>>	
使用“已知”表示，最新模块		将最新模块表示为“未知”模块
		

在SIMATIC管理器中使用代表模块进行工作

代表模块在站点层中是可见的。位于该层的所有附属对象，例如用户程序、系统数据和连接表，均是可见的，并可从 SIMATIC 管理器中下载。

您也可打开、编辑、编译和装载用户程序(例如用户程序的块)。

然而，对具有典型块的项目将有下列限制：

- 不能复制包含有典型块的站。
- 在菜单命令“项目另存为...”中，选项“重新组织”将不能完全适用。
  在复制和重新组织的项目(例如，用户程序)中，代表模块及该模块的所有引用和附属对象都将丢失。

在硬件组态中使用代表模块进行工作

代表模块显示在组态时所在的插槽上。

您可打开该模块，但不能改变其参数或为其下载参数。模块属性将限制为“典型”属性标签页中给定的那些属性。站组态不能修改(例如添加新的模块)。

也可进行硬件诊断(例如在线打开站) (有一定的限制：不能识别新的诊断选项和文本)。

在网络组态中使用代表模块进行工作

代表模块也可显示在 NetPro 中。此时，站上模块的名称将以问号开头。

具有代表模块的项目在 NetPro 中只能以写保护的形式打开。

在写保护模式下打开项目时，您可显示并打印网络组态。您也可获取连接状态，这些状态至少包括正在使用的 STEP 7 版本所支持的那些信息。

然而，通常情况下，您将无法对其进行任何修改、保存、编译或下载。

模块的后续安装

如果模块来自于较早版本的STEP 7，且有一个HW更新供其使用，则您可使用一个“真正的”模块来代替代表模块。一旦打开站点，您就将接收到与必需的HW更新或选项包有关的信息，且可使用对话框对其进行安装。另外，您也可选择菜单命令**选项 > 安装HW更新**来安装这些模块。

8 定义符号

8.1 绝对和符号寻址

在STEP 7程序中，使用地址如I/O信号、位存储器、计数器、定时器、数据块和功能块。完全可以在程序中访问这些地址，但是如果使用地址符号，程序将更容易阅读(例如，Motor_A_On或其它符合公司或行业内代码系统的符号)。然后，可以通过此符号访问用户程序中的地址。

绝对地址

绝对地址包含地址标识符和存储器位置(例如，Q 4.0、I 1.1、M 2.0、FB21)。

符号地址

如果将符号名分配给绝对地址，可以使程序更易读，并能简化故障查找。

STEP 7 可以自动将符号名翻译成所需要的绝对地址。如果愿意使用符号名称访问ARRAY、STRUCT、数据块、本地数据、逻辑块和用户自定义数据类型，在能用符号寻址数据前，必须首先将符号名称分配给绝对地址。

例如，可以将符号名称 MOTOR_ON 分配给地址 Q 4.0，然后在程序语句中将MOTOR_ON 作为地址使用。使用符号地址，更容易识别程序中的元素与过程控制项目的组件的匹配程度。

注意

符号名(变量ID)中不允许出现两个连续的下划线字符(例如，MOTOR__ON)。

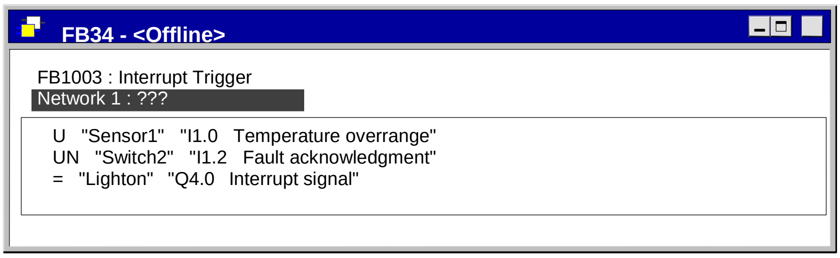
支持编程

在编程语言逻辑图、功能块图和语句表中，可以输入地址、参数和块名称，作为绝对地址或符号。

使用菜单命令*视图 > 显示 > 符号表示法*，可以在地址的绝对表示法和符号表示法之间切换。

为了更容易使用符号地址编程，可以显示绝对地址和属于符号的符号注释。可以使用菜单命令**视图 > 显示 > 符号信息**激活此信息。这意味着每个 STL 语句后的行注释中包含更多的信息。不能编辑显示；任何改变都必须在符号表或变量声明表中进行。

下图显示在 STL 中的符号信息。



当打印输出一个块时，具有语句注释或符号注释的当前画面表示也被打印。

8.2 共享符号与本地符号

符号使您能够采用具有某种意义的符号名来代替绝对地址进行工作。短符号和长注释的有效结合，可使编程更容易、程序文档的质量更好。

应注意区分局部(指定块)符号和共享符号之间的不同。

	共享符号	局部符号
有效性	• 在整个用户程序中均有效， • 所有块均可使用， • 在所有块中均具有相同含义， • 在整个用户程序中必须是唯一的。	• 仅在对其进行定义的块中有效； • 同一个符号可以根据不同用途在不同的块中使用。
允许的字符	• 字母、数字、特殊字符， • 除0x00、0xFF以外的重音符以及引号， • 如果使用特殊字符，则必须将其放置在引号内。	• 字母， • 数字， • 下划线(_)。

	共享符号	局部符号
使用方法	• 可定义共享符号用于： • I/O信号(I、IB、IW、ID、Q、QB、QW、QD) • I/O输入和输出(PI、PQ) • 位存储器(M、MB、MW、MD) • 定时器(T)/计数器(C) • 逻辑块(OB、FB、FC、SFB、SFC) • 数据块(DB) • 用户自定义的数据类型(UDT) • 变量表(VAT)	• 可定义局部符号用于： • 块参数 (输入、输出以及输入/输出参数)， • 块的静态数据， • 块的临时数据。
在何处定义？	符号表	块的变量声明表

8.3 显示共享符号或局部符号

程序代码段中的共享符号与本地符号之间的差别可区分如下：

- 来自符号表中的符号(共享符号)将显示在引号“..”内。
- 来自块的变量声明表中的符号(局部符号)将在前面冠以字符“#”。

引号或“#”无须输入。在梯形图、FBD 或 STL 中输入程序时，语法检查将自动添加这些字符。

如果担心在某些情况下出现混淆，例如在符号表和变量声明中都使用同一个符号，那么当您要使用该共享符号时，必须直接对其进行编码(输入地址或者包括引号的符号)。此时，没有进行分别编码的任何符号都将解释为指定块(局部)的变量。

如果符号包含有空格，也必须对共享符号进行编码(输入地址或者包括引号的符号)。

当在 STL 源文件中进行编程时，将采用同样的特殊字符及准则。在自由编辑模式下，将不会自动添加代码字符，但如果您希望避免混淆，这些代码字符将仍然需要。

注意

使用菜单命令**视图 > 显示 > 符号表达式**，可切换显示所声明的共享符号与绝对地址。

8.4 设置地址优先级(符号/绝对地址)

地址优先级有助于，在改变符号表中的符号、改变数据块或功能块的参数名称、改变指向组件名称的 UDT 或修改多重背景时，按照您的意愿调整程序代码。

当在下列情况下进行改变时，必须仔细设置地址优先级，并要有明确的目的。为了能对地址优先级善加利用，在您开始另一类型的改变之前，每个改变过程都必须彻底完成。

为了设置地址优先级，请跳转到SIMATIC管理器，并选择块文件夹，然后选择菜单命令**编辑 > 对象属性**。在“地址优先级”标签中，您可进行合适的设置。

对地址优先级的优化设置要求区分下列改变情况：

- 纠正单个名称
- 切换名称或分配
- 新的符号、变量、参数或组件

注意

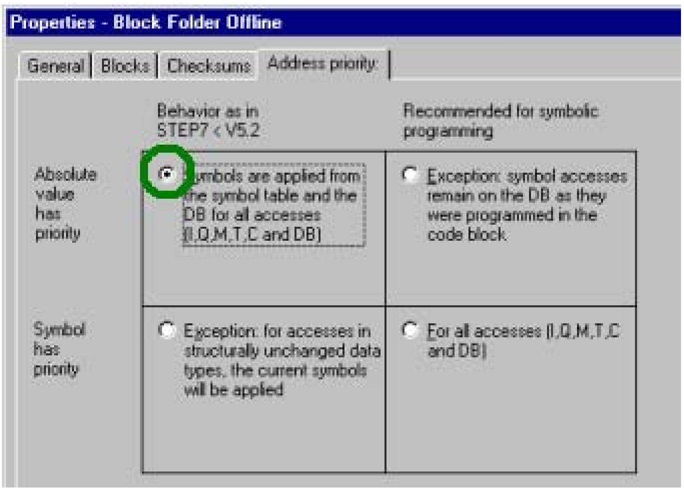
请记住，即使已经设置了符号地址优先级，绝对块编号仍是调用逻辑块时的决定因素(“调用FC”或“调用FB、DB”)!

纠正单个名称

实例：

在符号表或程序编辑器/块编辑器中，必须纠正名称的拼写错误。这适用于符号表中的所有名称，以及可使用程序编辑器/块编辑器进行修改的所有参数、变量或组件的名称。

设置地址优先级：



跟踪修改：

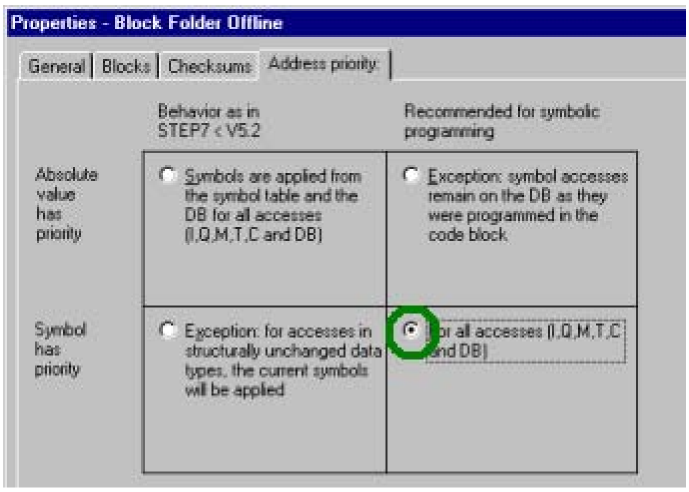
在SIMATIC管理器中，选择块文件夹，然后选择菜单命令**编辑 > 检查块一致性**。“检查块一致性”功能使得单个块中有必要进行修改。

切换名称或分配

实例：

- 符号表中现有分配的名称已切换。
- 符号表中现有的分配分配了新地址。
- 变量名称、参数名称或组件名称在程序编辑器/块编辑器中进行切换。

设置地址优先级：



跟踪修改：

在SIMATIC管理器中，选择块文件夹，然后选择菜单命令**编辑 > 检查块一致性**。“检查块一致性”功能在单个块中进行必要的修改。

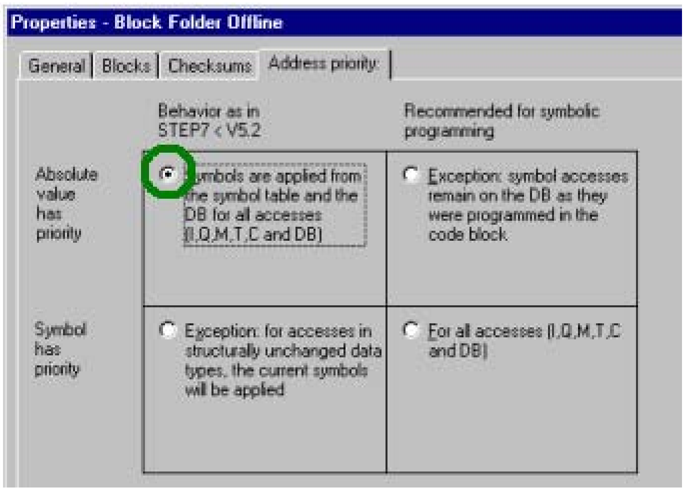
新的符号、变量、参数或组件

实例：

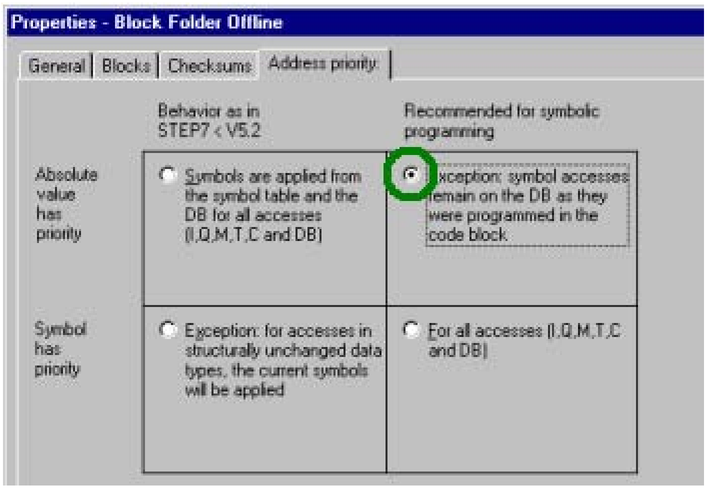
- 正在为程序中使用的地址创建新符号。
- 正在向数据块、UDT 或功能块中添加新的变量或参数。

设置地址优先级：

- 在符号表中改变。



- 在程序/块编辑器中改变。



跟踪修改：

在SIMATIC管理器中，选择块文件夹，然后选择菜单命令**编辑 > 检查块一致性**。“检查块一致性”功能使得单个块中有必要进行修改。

8.5 共享符号的符号表

8.5.1 共享符号的符号表

在符号表中定义共享符号。

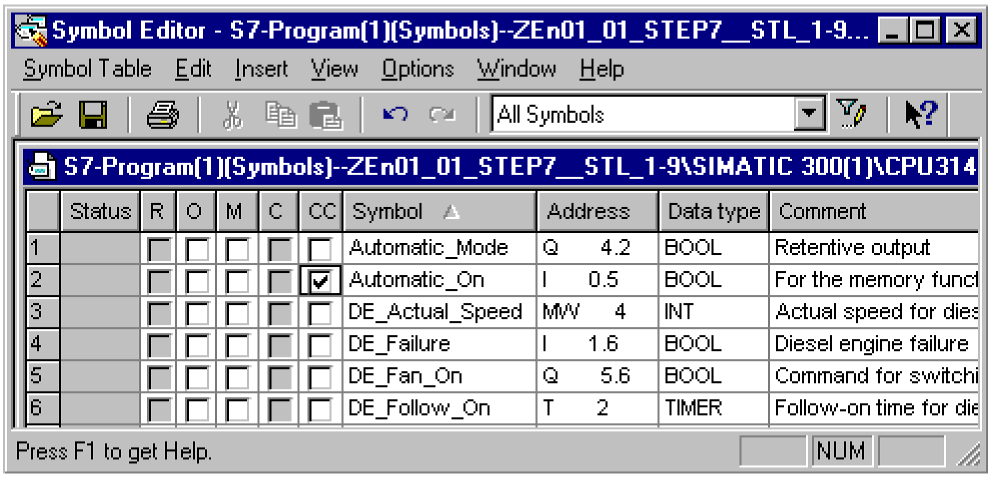
在创建 S7 或 M7 程序时，将自动创建一个(空的)符号表(“符号”对象)。

有效性

符号表仅适用于要为其链接程序的模块。如果要在众多不同的 CPU 中使用同样的符号，必须自己确保各种符号表中的条目全都匹配(例如，通过复制符号表)。

8.5.2 符号表的结构和组件

符号表的结构



行

	如果“特殊对象属性”列已隐藏(菜单命令**视图 > 列O、M、C、R、CC**已取消选定)，则该符号将出现在行中，只要相关的行至少为其设置了一个“特殊对象属性”。

“状态”列

=	符号名或地址与符号表中的另一个条目是一样的。
X	符号仍然是不完整的(符号名或地址已丢失)。

R/O/M/C/CC列

R/O/M/CC 列说明是否为符号分配了特殊的对象性质(属性)：

- R (监视)意味着使用选项包 S7-PDIAG (V5)为符号创建了过程诊断的错误定义。
- O 意味着可使用 WinCC 对符号进行操作和监视。
- M 意味着与符号相关的消息(SCAN)已经分配给符号。
- C 意味着符号已分配了通讯属性。
- CC 意味着可在程序编辑器中对符号进行快速、直接的监视和控制(“触点控制”)。

点击复选框，激活或禁止这些“特殊对象属性”。您也可通过**编辑 > 特殊对象属性**菜单命令来编辑“特殊对象属性”。

“符号”列

符号名不能长于 24 个字符。

不能在符号表中为数据块(DBD、DBW、DBB、DBX)地址分配符号。其名称在数据块声明中进行分配。

对于组织块(OB)和某些系统功能块(SFB)以及系统功能(SFC)，预先定义的符号表条目已经存在，在编辑 S7 程序的符号表时，您可将其导入表中。导入文件将存储在 STEP 7 目录...\S7data\符号\符号.sdf 中。

“地址”列

地址是特定存储区和存储单元的缩写。
实例：Input I 12.1

输入后，将对地址的语法进行检查。

“数据类型”列

您可在 STEP 7 的众多数据类型之间进行选择。数据类型域已经包含了缺省数据类型，必要时可进行改变。如果您的改变不适用于地址，或语法不正确，则当您从域中退出时，将显示一条出错消息。

“注释”列

您可为所有的符号分配注释。简短的符号名和更详细的注释相组合，会使程序创建更有效、程序文档更完整。注释长度多达 80 个字符。

转换为C变量

您可从 M7 程序的符号表中选择符号，并结合 ProC/C++软件选项，将其转换为相应的 C 变量。

8.5.3 符号表中所允许的地址和数据类型

在整个符号表中仅可使用一套助记符。在SIMTIC管理器中，必须使用菜单命令**选项 > 自定义**在“语言”标签中进行SIMATIC (德语)和IEC (英语)助记符之间的切换。

IEC	SIMATIC	描述	数据类型	地址范围
I	E	输入位	BOOL	0.0至65535.7
IB	EB	输入字节	BYTE, CHAR	0至65535
IW	EW	输入字	WORD, INT, S5TIME, DATE	0至65534
ID	ED	输入双字	DWORD, DINT, REAL, TOD, TIME	0至65532
Q	A	输出位	BOOL	0.0至65535.7
QB	AB	输出字节	BYTE, CHAR	0至65535
QW	AW	输出字	WORD, INT, S5TIME, DATE	0至65534
QD	AD	输出双字	DWORD, DINT, REAL, TOD, TIME	0至65532
M	M	存储器位	BOOL	0.0至65535.7
MB	MB	存储器字节	BYTE, CHAR	0至65535
MW	MW	存储器字	WORD, INT, S5TIME, DATE	0至65534
MD	MD	存储器双字	DWORD, DINT, REAL, TOD, TIME	0至65532
PIB	PEB	外设输入字节	BYTE, CHAR	0至65535
PQB	PAB	外设输出字节	BYTE, CHAR	0至65535
PIW	PEW	外设输入单字	WORD, INT, S5TIME, DATE	0至65534
PQW	PAW	外设输入双字	WORD, INT, S5TIME, DATE	0至65534
PID	PED	外设输入双字	DWORD, DINT, REAL, TOD, TIME	0至65532
PQD	PAD	外设输出双字	DWORD, DINT, REAL, TOD, TIME	0至65532
T	T	定时器	TIMER	0至65535
C	Z	计数器	COUNTER	0至65535
FB	FB	功能块	FB	0至65535
OB	OB	组织块	OB	1至65535
DB	DB	数据块	DB, FB, SFB, UDT	1至65535
FC	FC	功能	FC	0至65535
SFB	SFB	系统功能块	SFB	0至65535
SFC	SFC	系统功能	SFC	0至65535
VAT	VAT	变量表		0至65535
UDT	UDT	用户自定义的数据类型	UDT	0至65535

8.5.4 符号表中的不完全和非唯一符号

不完全符号

也可以存储不完全符号。例如，可以先输入符号名，然后再在以后添加相应的地址。这意味着可随时中断对符号表的操作、保存中间结果，然后在另外的时间内完成其余的工作。不完全符号在“状态”栏中将用符号进行标识。在准备使用符号编写软件(没有错误消息出现)时，必须输入符号名、地址和数据类型。

多义符号是如何产生的

在将符号插入到符号表，而其符号名和/或地址已用于另一符号栏时，就会产生多义符号。这意味着新符号和现有符号都是不确定的。“状态”栏中的符号表明了这种状态。

例如，在复制和粘贴某个符号以便对副本中的详细情况进行略微的修改时，就将发生这种情况。

多义符号的标识

在符号表中，多义符号可通过图形方式(颜色、字体)加亮显示来进行标识。其表达式中的这种变化意味着它们仍然需要编辑。您既可显示所有符号，也可对视图进行过滤，以便只显示唯一符号或多义符号。

使符号唯一

如果改变了导致这种状态的组件(符号和/或地址)，多义符号将变为唯一符号。如果两个符号都是不确定的，并且您已改变了其中的一个符号以使其成为唯一符号，那么另一个符号也将变为唯一符号。

8.6 输入共享符号

8.6.1 输入共享符号

在随后的阶段中，有三种输入符号的方法可供编程使用：

- 通过对话框
  在正在输入程序的窗口中打开一个对话框，然后定义一个新的符号或重新定义现有的符号。建议在定义单个的符号时使用该过程，例如，当您意识到符号可能丢失或希望在编写程序时修改符号的时候使用。这样可以不必显示整个符号表。
- 直接在符号表中
  可直接在符号表中输入符号及其绝对地址。如果希望输入许多符号，或者当您为了使已分配的符号在屏幕上显示而创建项目的符号表时，建议使用该过程，它可容易地对符号进行总览。
- 从其它表格编辑器中导入符号表
  可在您熟知的任何表格编辑器(例如Microsoft Excel)中创建符号表的数据，然后将所创建的文件导入符号表。

8.6.2 与输入符号有关的常用提示

要为符号表输入新的符号，可将光标放置在表中的第一个空白行，并对单元进行填充。可使用菜单命令**插入 > 符号**将新行插入到符号表当前行的前面。如果在光标位置之前的行已经包含了地址，则可以通过预设“地址”栏和“数据类型”栏来插入新符号。地址来自前一行，根据前一行自动添加；数据类型则输入缺省的数据类型。

使用编辑菜单中的命令可复制和修改现有的条目。进行保存，然后关闭符号表。也可保存尚未完全定义的符号。

在输入符号时，应注意以下要点：

列	注意
符号	整个符号表内的名称必须是唯一的。当对该域中的条目进行确认或退出域时，将对非唯一符号进行标记。符号最多可包含24个字符。不可使用引号(")。
地址	当对该域中的条目进行确认或退出域时，将检查输入的地址是否合法。
数据类型	当输入地址时，将自动为该域分配一个缺省的数据类型。如果改变该缺省设置，程序将检查新的数据类型是否与地址匹配。
注释	可在此处输入注释，以简要介绍符号的功能(最多80个字符)。输入注释是可选的。

8.6.3 在对话框中输入单个共享符号

下述步骤说明了当对块进行编程时，如何不显示符号表就能在对话框中修改符号或定义新符号。

如果只是希望编辑单个符号，该步骤十分有用。如果想要编辑多个符号，那么应打开符号表并直接在符号表中工作。

激活块中的符号显示

使用菜单命令**视图 > 显示 > 符号表达式**在打开块的块窗口中激活符号显示。菜单命令前面将出现复选标记，表示符号显示已激活。

在输入程序时定义符号

1. 确保块窗口中的符号表达式已打开(菜单命令**视图 > 显示 > 符号表达式**。)
2. 在想要为其分配符号的程序代码段中，选择绝对地址。
3. 选择菜单命令**编辑 > 符号**。
4. 填写对话框然后将其关闭，单击“确定”确认您的输入并确保输入了一个符号。

所定义的符号将输入到符号表中。如果所作的输入会导致出现非唯一性符号，则会出现出错消息并将其拒绝。

在符号表中编辑

使用菜单命令**选项 > 符号表**，可打开符号表以进行编辑。

8.6.4 在符号表中输入多个共享符号

打开符号表

可使用多种方法打开符号表：

- 双击项目窗口中的符号表。
- 选择项目窗口中的符号表，并选择菜单命令**编辑 > 打开对象**。

活动程序的符号表将显示在它自己的窗口中。现在即可创建符号或对其进行编辑。当符号表创建之后首次打开时，符号表是空白的。

输入符号

要为符号表输入新的符号，可将光标放置在表中的第一个空白行，并对单元进行填充。可使用菜单命令**插入 > 符号**将新的空白行插入到符号表当前行的前面。使用编辑菜单中的命令可复制和修改现有的条目。进行保存，然后关闭符号表。也可保存尚未完全定义的符号。

排序符号

符号表中的数据记录可根据符号、地址、数据类型、或注释，按字母表顺序进行排序。

使用菜单命令**视图 > 排序**打开对话框并定义经排序的视图，可改变符号表的排序方式。

过滤符号

可使用一个过滤器来选择符号表中记录的子集。

使用菜单命令**视图 > 过滤器**可打开“过滤器”对话框。

您可定义记录必须满足的标准，以便将其包括在经过滤的视图中。过滤可根据：

- 符号名、地址、数据类型、注释
- 具有操作员监控属性的符号、具有通讯属性的符号、消息二进制变量的符号(位存取器或过程输入)
- 具有状态“有效”、“无效(非唯一、未完成)”的符号

各个标准均链接有一个“与”操作。经过滤的记录将从指定的字符串开始。

如果您希望了解“过滤器”对话框中有关选项的更多信息，请按下 F1 打开前后文相关的在线帮助。

8.6.5 使用大写和小写符号

在大写和小写字符之间没有任何区别

以前可以在 STEP 7 中定义一些符号，这些符号在用于单独的字符时仅在大小写上有区别。在 STEP 7 的 V4.02 中这种情况发生了变化。现在将不再按照大小写对符号进行区分。
该变化是根据我们客户的要求而作出的，它将大大减少程序中出现错误的风险性。对符号定义作出的这些限制也支持 PLC 开放论坛的目标，即为可转换的程序定义一个标准。

现在已经不再支持仅按照大小写字符进行区分的符号定义。例如，以前在符号表中可进行如下定义：

Motor1 = I 0.0

motor1 = I 1.0

符号将根据第一个字母所使用的大小写情况进行区分。这类区别具有容易混淆的危险。新的定义将消除此错误来源。

对现有程序的影响

如果已经使用该标准来区分不同的符号，那么在下列情况下使用新定义时可能会遇到困难：

- 符号**仅**在使用了大小写字符时才进行相互区分
- 参数**仅**在使用了大小写字符时才进行相互区分
- 符号**仅**在使用了大小写字符时才与参数进行区分

然而，所有的这三种冲突都可通过下述方法进行分析和处理。

仅在使用了大小写字符时才进行相互区分的符号

冲突：

如果没有用当前版本的软件对符号表进行编辑，那么在编译源文件时将使用符号表中第一个非唯一的符号。

如果已经对符号表进行了编辑，那么这类符号将无效；也就是说在打开块时将不显示符号，并且包含这些符号的源文件在编译时可能会出错。

纠正方法：

打开符号表，检查其中是否存在冲突，然后重新保存符号表。该动作可识别出非唯一的符号。随后，即可使用过滤器“非唯一的符号”来显示非唯一的符号，然后对其进行更正。还应该更正包含冲突的任何源文件。您无须对块进行更多的修改，因为在打开块时将自动使用或显示当前(现在没有冲突)版本的符号表。

仅在使用了大小写字符时才进行相互区分的参数

冲突：

包含这类接口的源文件在编译时可能会出错。可打开具有这类接口的块，但不能再访问这些参数中的第二个参数。当试图访问第二个参数时，程序将自动返回到保存块时的第一个参数。

纠正方法：

要检查哪些块包含有这类冲突，建议使用功能“生成源文件”生成一个用于程序所有块的源文件。如果在试图编译已经创建的源文件时出现错误，那么肯定存在冲突。

通过确保参数唯一，例如使用“查找和替换”功能，可更正源文件。然后再次对源文件进行编译。

仅在使用了大小写字符时才与参数进行区分的符号

冲突：

如果源文件中的共享符号和局部符号仅在使用了大小写字符时才进行相互区分，而且没有使用任何初始字符来识别共享(“符号名”)或局部(#符号名)符号，那么在编译期间将始终使用局部符号。这将导致机器代码被修改。

纠正方法：

此时，建议生成一个适用于所有块的新的源文件。这将自动分配具有相应初始字符的局部和共享访问，并将确保在今后的编译过程期间对其进行正确处理。

8.6.6 导出和导入符号表

可将当前的符号表导出到一个文本文件中，以便能够使用任意的文本编辑器对其进行编辑。

也可以将使用另一个应用程序创建的表格导入到您的符号表中，然后继续在那里编辑。例如，导入功能可用于将使用 STEP5/ST 创建的设置列表转换后导入到符号表中。

可供选择的文件格式有*.SDF、*.ASC、*.DIF 和*.SEQ。

导出规则

可导出整个符号表、已过滤的符号表子集或符号表视图中的所选行。

可以使用菜单命令**编辑 > 特殊对象属性**，将不导出设置的符号属性。

导入规则

- 对于常用的系统功能块(SFB)、系统功能(SFC)和组织块(OB)，在文件 ...\S7DATA\SYMBOL\SYMBOL.SDF 中提供了预定义的符号表条目，可根据需要导入。
- 在导出和导入时，将不考虑使用菜单命令**编辑 > 特殊对象属性**设置的符号属性。

8.6.7 用于导入/导出符号表的文件格式

下列文件格式均可导入到符号表或从符号表中导出：

- ASCII 文件格式(ASC)
- 数据交换格式(DIF)
  您可在Microsoft Excel中打开、编辑和保存DIF文件。
- 系统数据格式(SDF)
  您可在Microsoft Access中打开、编辑和保存SDF文件。
  - 为在 Microsoft Access 应用程序中对数据进行导入和导出，可使用 SDF 文件格式。
  - 在 Access 中，选择文件格式“文本(带分隔符)”。
  - 使用双引号(") 作为文本分隔符。
  - 使用逗号(,) 作为单元格分隔符。
- 设置列表(SEQ)
  警告：在将符号表导出到类型为.SEQ的文件时，长度大于40个字符的注释将截去第40个字符之后的部分。

ASCII 文件格式(ASC)

文件类型	*.ASC
结构：	记录长度，分隔符逗号，记录
实例：	126,green_phase_ped. T 2 TIMER 人行横道绿灯持续时间 126,red_ped. Q 0.0 BOOL 人行横道红灯

数据交换格式(DIF)

文件类型	*.DIF
结构：	DIF文件由文件头和数据组成：

报头	表	DIF文件的起始部分
	0.1	
	"<标题>"	注释字符串
	VECTORS	文件中的记录数
	0,<记录数>	
	""	
	TUPLES	记录中数据域的数目
	0,<列数>	
	""	
	DATA	用于文件头结尾和数据起始部分的标识号
	0,0	
	""	
数据(每条记录)	<类型>,<数字值>	用于数据类型、数值的标识号
	<字符串>	字母数字部分或
	V	如果未使用字母数字部分

文件头：文件头必须按指定次序包含记录类型TABLE、VECTORS、TUPLES以及DATA。在DATA前面，DIF文件可包含更多可选的记录类型。然而，符号编辑器将忽略这些类型。
数据：在数据部分，每个条目由三部分组成：类型(数据类型)的标识号、数值以及字母数字部分。

您可在 Microsoft Excel 中打开、编辑和保存 DIF 文件。不要使用重音符、元音变音或其它特殊语言字符。

系统数据格式(SDF)

文件类型	***.SDF**
结构：	引号中的字符串，用逗号隔开的部分
实例：	"green_phase_ped.","T 2","TIMER","人行横道绿灯持续时间" "red_ped.","Q 0.0","BOOL","人行横道红灯"

要在 Microsoft Access 中打开 SDF 文件，应选择文件格式‘文本(带分隔符)’。使用双引号(")作为文本分隔符，使用逗号(,)作为域分隔符。

设置列表(SEQ)

文件类型	***.SEQ**
结构：	TAB 地址 TAB 符号 TAB 注释 CR
实例：	T 2 green_phase_ped.人行横道绿灯持续时间 Q 0.0 red_ped.人行横道红灯

TAB 表示制表键(09H)，CR 表示 RETURN 回车键(0DH)。

8.6.8 符号表中的编辑区

从STEP 7 V5.3版本起，可在符号表中选择并编辑连续区域。这表示可复制和/或剪切部分符号表，并将其插入到另一个符号表中，或按要求将其删除。
通过将数据从一个符号表快速传送到另一个符号表，更易于更新符号表。

可选择的区域：

一点击行中第一列，就可选择整行。如果希望选择从“状态”列到“备注”列的所有域，那么这些也是所选择行的一部分。

- 可选择一个或多个连续域作为整个区。为了能选择该区，所有域必须属于“符号”、“地址”、“数据类型”和“备注”列。如果做了无效选择，那么不能使用编辑菜单命令。
- R、O、M、C、CC 列包含各符号的特殊对象属性，并只有在选择了“自定义”对话框(菜单命令**选项 > 自定义**)中的“还复制特殊对象属性”复选框时，才会复制这些列。
- 如果显示 R、O、M、C、CC 列，那么复制这些列的内容。要显示或隐藏这些列，请选择**视图 > R、O、M、C、CC 列**菜单命令。

要编辑符号表，请按如下进行操作：

1. 使用下列两种方法之一，可选择要在符号表中编辑的区：
   - 用**鼠标**点击起始单元，按住鼠标左键，将鼠标移到要选择的区。
   - 通过**键盘**，按住 shift 键，再按光标(箭头)键，选择区。
2. 选定的区反白显示。第一个选中的单元以正常方式显示，周围有一个边框。
3. 按要求编辑选定的区。

9 创建块和库

9.1 选择编辑方法

根据创建程序时所使用的编程语言，在增量输入模式和/或自由编辑(文本)模式下都可输入程序。

用于梯形图(LAD)、功能块图(FBD)、语句表(STL)、或 S7-GRAPH 编程语言的增量编辑器

将在用于 LAD、FBD、STL 和 S7-GRAPH 的增量输入模式编辑器中，创建存储在用户程序中的块。如果希望立即检查刚才已输入的内容，应选择使用增量输入模式。该编辑模式尤其适用于初学者。在增量输入模式中，当每行或每个元素均输入完毕之后，将立即对其进行语法检查。在编辑语句之前，必须对所使用的符号进行定义。语法正确的输入项将自动进行编译，并存储在用户程序中。

在编辑语句之前，所使用的符号必须已经定义。如果没有可供使用的符号，则块将不能完整地进行编译；但是，可以保存这种不一致的中间版本。

用于 STL、S7 SCL、或 S7 HiGraph 等编程语言的源代码(文本)编辑器

在源代码编辑器中，将创建用于后续编译的**源代码文件**，以生成块。

建议使用源代码编辑器，因为这是一个非常有效的对程序进行编辑和监视的方法。

将在文本文件中编辑程序或块的源代码，然后进行编译。

文本文件(源文件)将存储在S7程序的源文件夹中，例如，存为**STL 源文件**或**SCL 源文件**。源文件可包含一个或多个块的代码。STL和SCL文本编辑器可以生成**OB**、**FB**、**FC**、**DB**、**以及UDT** (用户自定义数据类型)的源代码，这样，您可使用它们来创建完整的用户程序。一个这样的文本文件可能包含完整的CPU程序(即所有的块)。

编译源文件时将生成相应的块，并写入用户程序中。在对其进行编译之前，必须定义所有使用的符号。且不报告数据错误，直到相应的编译器编译完源文件为止。

编译必须符合编程语言的规定语法。只有根据用户指令或在源文件被编译成块时，才执行语法检查。

9.2 选择编程语言

设置编辑器的编程语言

在生成特定块或源文件之前，可通过对象属性选择编程语言和编辑器。这种选择将确定打开块或源文件时将启动哪一个编辑器。

启动编辑器

在SIMATIC管理器中双击相应的对象(块、源文件等)，或者，选择菜单命令**编辑 > 打开对象**或单击相应的工具栏按钮，都可以启动合适的语言编辑器。

为创建 S7 程序，表中列出的编程语言均可供使用。标准 STEP 7 软件包提供有 STEP 7 编程语言 LAD、FBD 以及 STL。可按选项软件包购买其它的编程语言。

然后即可选择多种不同的编程方法(梯形图、功能块图、语句表、标准语言、顺序控制、或状态图)并选择是使用基于文本的编程语言，还是图形编程语言。

选择一种编程语言以确定输入模式(•)。

编程语言	用户组	应用	增量输入	自由编辑模式	块可从CPU重新归档
语句表STL	偏好使用类似机器码的语言进行编程的用户	程序将根据运行时间和存储器要求进行优化	•	•	•
梯形图LAD	习惯于使用电路图进行工作的用户	编写逻辑控制程序	•	–	•
功能块图FBD	熟练布尔代数的逻辑框的用户	编写逻辑控制程序	•	–	•
F-LAD、F-FBD 选项包	熟悉编程语言LAD和FDB的用户	编写F系统的安全程序	•	–	•
SCL (结构控制语言) 选项包	使用过高级语言例如PASCAL或C语言进行编程的用户	编写数据处理任务程序	–	•	–

编程语言	用户组	应用	增量输入	自由编辑模式	块可从**CPU**重新归档
S7-GRAPH 选项包	希望面向技术功能进行工作且不具备丰富编程/PLC知识的用户	顺序控制的简便描述	•	–	•
HiGraph 选项包	希望面向技术功能进行工作且不具备丰富编程/PLC知识的用户	异步、非顺序控制的简便描述	–	•	–
CFC 选项包	希望面向技术功能进行工作且不具备丰富编程/PLC经验的用户	连续过程的描述	–	–	–

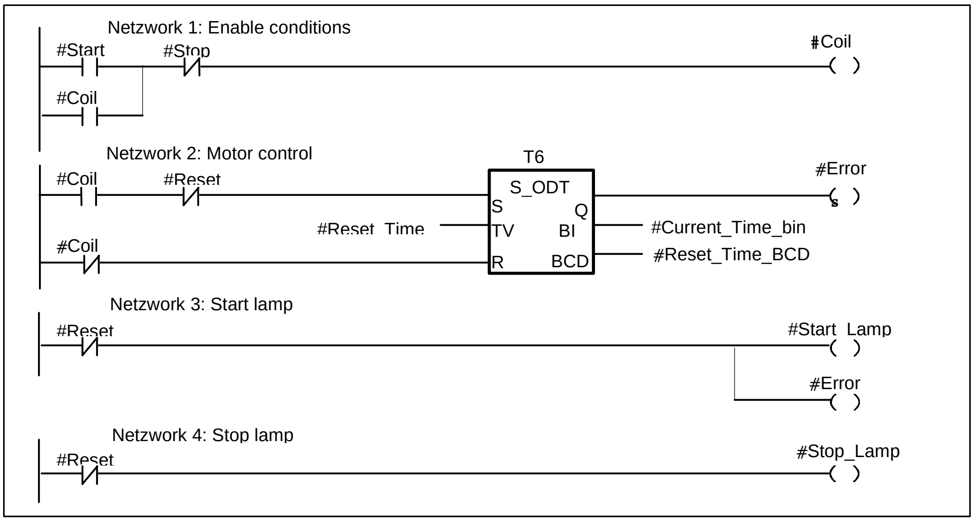
如果块中没有任何错误，则可在梯形图、功能块图、或语句表等格式之间切换。不能在目标语言下显示的程序部分将用语句表格式来显示。。

在 STL 下，可通过源文件来生成块，反之亦然。

9.2.1 梯形图编程语言(LAD)

图形编程语言“梯形图(LAD)”以电路图表示为基础。电路图的元件，例如常开触点和常闭触点，相互组合，从而构成程序段。逻辑块的代码段表示一个或多个程序段。

LAD 程序段实例



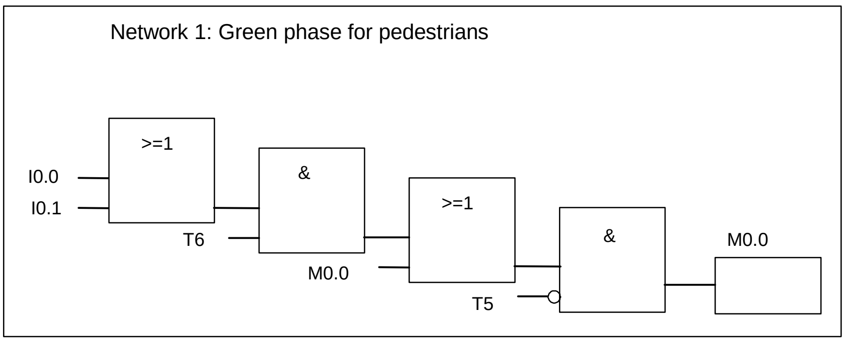
编程语言LAD将随标准STEP 7软件包提供。使用增量编辑器可创建LAD程序。

9.2.2 功能块图编程语言(FBD)

编程语言“功能块图(FBD)”以布尔代数中众所周知的图形逻辑符号为基础。也可以通过逻辑框的组合直接显示诸如数学函数的复杂功能。

编程语言 FBD 将随标准 STEP 7 软件包提供。

FBD 程序段实例



使用增量编辑器可创建 FBD 程序。

9.2.3 语句表编程语言(STL)

编程语言STL是一种基于文本的编程语言，它具有一种类似机器代码的结构。每个语句代表CPU的一种程序处理操作。多个语句链接在一起就构成了程序段。

语句表程序段的实例

```
        Network 1: Control drain valve
A (
O
O   #Coil
)
AN  #Close
=   #Coil
        Network 2: Display "Valve open"
A  #Coil
=   #Disp_open
          Network 3: Display "Valve closed"
AN  #Coil
=   #Disp_closed
```

编程语言STL将随标准STEP 7软件包提供。通过该编程语言，您可使用增量编辑器编辑S7块，并可在源代码编辑器中创建和编译STL程序源文件以生成块。

9.2.4 S7 SCL编程语言

编程语言 SCL (结构化控制语言)将作为选项包提供。这是一种基于文本的高级语言，其全局语言定义符合 IEC 1131-3。与 PASCAL 近似但不同于 STL 的这种语言，由于具有高级命令，将简化诸如循环和条件分支的编程。因此，SCL 适合于方程、复杂优化算法、或大规模数据管理等的计算。

S7 SCL 程序使用源代码编辑器进行编写。

实例：

```
FUNCTION_BLOCK FB20
VAR_INPUT
ENDVAL:                 INT;
END_VAR
VAR_IN_OUT
IQ1 ：          REAL;
END_VAR
VAR
INDEX :                 INT;
END_VAR

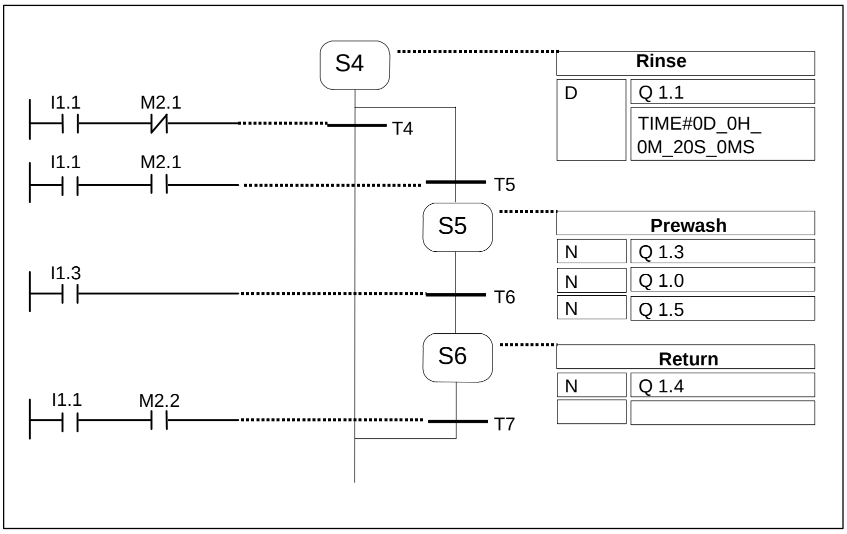
BEGIN
CONTROL:=FALSE;
FOR INDEX:= 1 TO ENDVALUE DO
    IQ1:= IQ1 * 2;
    IF IQ1 >10000 THEN
            CONTROL = TRUE
    END_IF
END_FOR;
END_FUNCTION_BLOCK
```

9.2.5 S7-GRAPH编程语言(顺序控制)

图形编程语言 S7-GRAPH 将作为选项包提供。它允许对顺序控制进行编程。这包括创建序列发生器以及指定相应的步内容与跳转。将在特定编程语言(类似于 STL)中对步的内容进行编程。在梯形图编辑器(LAD 的简化版)中对跳转进行编程。

S7-GRAPH 非常清楚地显示了特别复杂的序列，并使编程和疑难解答更为有效。

S7-GRAPH 下顺序控制的实例



所创建的块

使用S7-GRAPH编辑器，可对包含有序列发生器的功能块进行编程。相应的背景数据块包含有序列发生器的数据，例如，FB参数、步条件与跳转条件等。在S7-GRAPH编辑器中可自动生成该背景数据块。

源文件

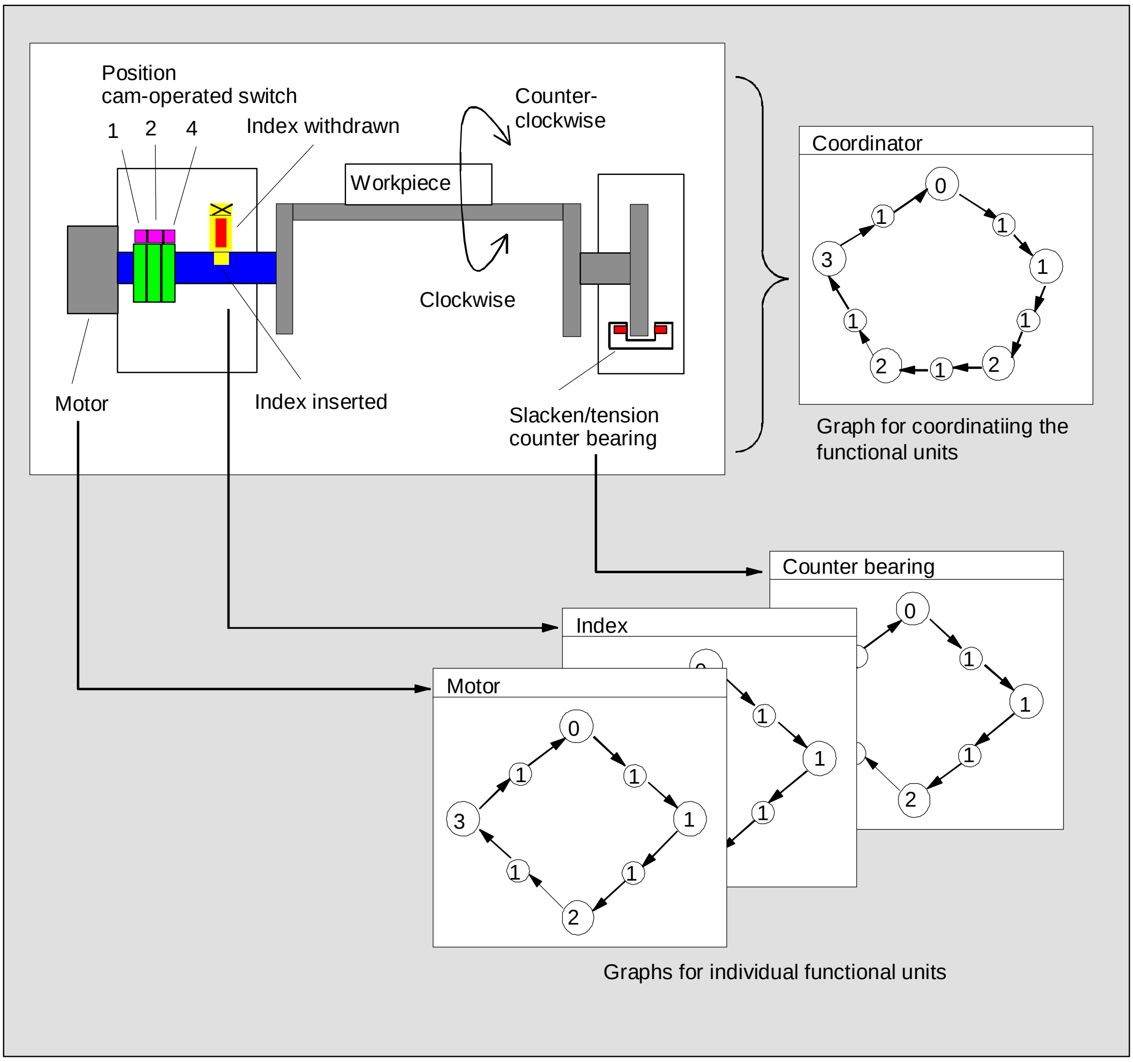
通过可由 OP 对其进行解释的 S7-GRAPH 所创建的功能块，或用于显示序列发生器的基于文本的显示，都可生成基于文本的源文件(GRAPH 源文件)。

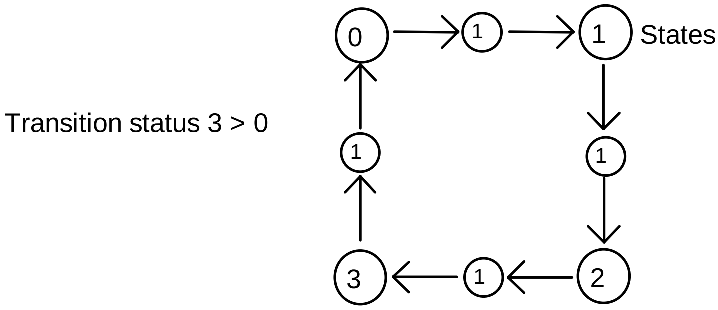
9.2.6 S7 HiGraph编程语言(状态图)

图形编程语言S7 HiGraph将作为选项包提供。它允许按状态图对程序中的许多块进行编程。在此，将把您的系统拆分为可获取不同状态的多个专用功能单元，并定义各种状态之间的转换条件。可使用类似于语句表的缩放型语言来描述分配给状态的动作以及状态之间的转换条件。

您可为每个功能单元创建一个描述该功能单元响应的图。各图组组合起来就构成了设备图。图之间可进行通讯，以对功能单元进行同步。

经合理安排的功能单元的状态转换视图，将使您能够进行系统编程并简化调试。S7-GRAPH 与 S7-HiGraph 之间差异在于：后者每一时刻仅获取一个状态(在 S7-GRAPH“步”中)。下图表示如何创建功能单元图(实例)。





图组将存储在 S7 程序“Source”文件夹中的 HiGraph 源文件中。于于是将对该源文件进行编译，以生成用于用户程序的 S7 块。

在完成图中的最后一次输入后将进行语法和格式检查(在工作窗口关闭时)。直到编译源文件时，才检查地址和符号信息。

9.2.7 S7 CFC编程语言

可选软件包 CFC (*Continuous Function Chart，连续功能图*)是一种编程语言，用于以图形化方式链接复杂功能。

使用编程语言 S7 CFC 可链接现有的各种功能。不需要自己对许多标准功能进行编程，取而代之的是您可以使用包含有标准块的库(例如，用于逻辑、数学运算、控制、以及数据处理等功能的库)。使用 CFC，您不需要任何详细的编程知识或可编程控制的专门知识，您只需将精力集中在您的专业领域所使用的技术上。

所创建的程序将以 CFC 图的形式进行存储。这些图均存储在 S7 程序下的“Charts”文件夹中。随后将对这些图进行编译，以形成用户程序的 S7 块。

您可能希望自己创建块并将其连接，此时，您可使用某种 S7 编程语言对 SIMATIC S7 的块进行编程，也可使用 C/C++对 SIMATIC M7 的块进行编程。

9.3 创建块

9.3.1 块文件夹

可按下面的形式创建 S7 CPU 的程序：

- 块
- 源文件

可使用 S7 程序下的文件夹“Blocks”来存储块。

该块文件夹包含有完成自动化任务而需要下载给 S7 CPU 的块。这些可装载的块包括逻辑块(OB、FB、FC)和数据块(DB)。在块文件夹中将自动创建一个空的组织块 OB1，因为在执行 S7 CPU 中的程序时将始终需要这个块。

块文件夹还包含有下列对象：

- 创建的用户自定义数据类型(UDT)。这些类型将使编程更容易，且不需要将其下载给 CPU。
- 为在调试程序时对变量进行监视和修改而创建的变量表(VAT)。不需要将变量表下载给 CPU。
- 包含有系统信息(系统组态、系统参数等)的对象“系统数据”(系统数据块)。在组态硬件时将创建并提供这些系统数据块。
- 在用户程序中需要调用的系统功能(SFC)与系统功能块(SFB)。您自己不能编辑 SFC 与 SFB。

除了系统数据块(只能通过可编程控制器的组态对其进行创建和编辑)，用户程序中的块都要使用各自的编辑器进行编辑。通过双击相应块即可启动对应的编辑器。

注意

按源文件编写，然后再进行编译的块，也将存储在块文件夹中。

9.3.2 用户自定义数据类型(UDT)

用户自定义数据类型是您自己创建的特定数据结构，一旦对其进行了定义，即可在整个 S7 程序中使用。

- 用户自定义数据类型既可像基本数据类型或复杂数据类型一样用于逻辑块(FC、FB、OB)的变量声明中，也可以用作数据块(DB)中的变量数据类型。其优点就是，您只需对特定数据结构定义一次，就能可以按照您的希望任意多次使用，并给它分配任意数目的变量。
- 用户自定义数据类型可当作一个模板，用于创建具有同一数据结构的数据块，这意味着您创建结构一次，以后就只需通过分配用户自定义数据类型来创建所需要的数据块(实例：配方：数据块的结构始终是相同的，仅使用的数量不同。)

正如其它块一样，在 SIMATIC 管理器或增量编辑器中都可以创建用户自定义数据类型。

用户自定义数据类型的结构

在打开一个用户自定义数据类型时，将显示一个新的工作窗口，该窗口将以表格的形式显示该用户自定义数据类型的声明视图。

- 第一行和最后一行已经包含有用于用户自定义数据类型起始和结束的声明 STRUCT 和 END_STRUCT。您不能编辑这两行。
- 从声明表的第二行开始，您即可通过在各列中输入您的条目来编辑用户自定义数据类型。
- 对用户自定义数据类型进行结构化，可根据：
  - 基本数据类型
  - 复杂数据类型
  - 现有的用户自定义数据类型

S7用户程序中的用户自定义数据类型将不下载给S7 CPU。即可直接使用增量输入编辑器先创建它们，然后进行编辑，也可在编译源文件时创建它们。

9.3.3 块属性

如果使用块属性，可以更容易地识别您创建的块，还可以保护这些块免受未授权地更改。

当块打开时，可编辑块属性。除了您可编辑的属性以外，属性对话框也将显示作为信息的数据，只是：您不能编辑该信息。

块属性和系统属性也将显示在 SIMATIC 管理器中块的对象属性中。此处，您只能编辑属性 NAME、FAMILY、AUTHOR 和 VERSION。

当您通过 SIMATIC 管理器插入块之后，您可编辑对象属性。如果使用某一不在 SIMATIC 管理器中的编辑器来创建块，则这些条目(程序语言)均将自动保存在对象属性中。

注意

在SIMATIC管理器中使用菜单命令**选项 > 自定义**和“语言”标签来设置用于S7块编程的那些助记符。

块属性表

在输入块属性时，应遵循下表中说明的输入顺序：

关键字/属性	含义	实例
[KNOW_HOW_PROTECT]	块保护；使用此选项编译的块将不能查看其代码段。可以查看块的接口，但不能更改。	KNOW_HOW_PROTECT
[AUTHOR:]	作者名：公司名、部门名或其它名称 (最多8个不含空格的字符)	AUTHOR：西门子公司，但无关键字
[FAMILY:]	块系列的名称：例如，控制器 (最多8个不含空格的字符)	FAMILY：控制器，但无关键字
[NAME:]	块名称(最多8个字符)	NAME：PID，但无关键字
[VERSION:int1 . int2]	块的版本号 (两个数都介于0和15之间，即0.0至15.15)	VERSION : 3.10
[CODE_VERSION1]	指示功能块是否可声明多重背景的标识号。如果想声明多重背景，则功能块不可具有此属性	CODE_VERSION1

关键字/属性	含义	实例
[UNLINKED]仅适用于DB	具有UNLINKED属性的数据块只存储在装入存储器中。它们不占用任何工作存储器空间，并且不与程序链接。不能使用MC7命令访问它们。此类DB的内容只能使用SFC 20 BLKMOV (S7-300. S7-400)或SFC 83 READ_DBL (S7-300C)传送给工作存储器。	
[Non-Retain]	具有该属性的数据块在每次掉电和上电之后以及CPU的每次STOP-RUN转换之后均将复位为装载值。	
[READ_ONLY]仅适用于DB	用于数据块的写保护；其数据只能读取，不能修改	READ_ONLY

块保护 KNOW_HOW_PROTECT 具有下列作用：

- 如果想在稍后阶段在 STL、FBD 或梯形图增量编辑器中查看已编译的块，将无法显示块的代码段。
- 块的变量声明表将只显示声明类型为 var_in、var_out 和 var_in_out 的变量。声明类型为 var_stat 和 var_temp 的变量保持隐藏状态。

分配：块属性给块类型

下表说明块类型及其对应的可声明块属性：

属性	OB	FB	FC	DB	UDT
KNOW_HOW_PROTECT	●	●	●	●	–
AUTHOR	●	●	●	●	–
FAMILY	●	●	●	●	–
NAME	●	●	●	●	–
VERSION	●	●	●	●	–
UNLINKED	–	–	–	●	–
READ_ONLY	–	–	–	●	–
不保持	–	–	–	●	–

KNOW_HOW_PROTECT 属性可在对块进行编程时，在源文件中进行设置。它将显示在“块属性”对话框中，但不能进行修改。

9.3.4 显示块长

块长度将按“字节”进行显示。

块文件夹属性中的显示

下列长度均将显示在离线视图中的块文件夹属性中：

- 可编程控制器的装入存储器大小(不带系统数据的所有块的总和)
- 可编程控制器的工作存储器大小(不带系统数据的所有块的总和)
- 可编程设备(PG/PC)上的块长度将不显示在块文件夹属性中。

块属性中的显示

以下均将显示在块属性中：

- 本地数据的所需数目：以字节为单位的本地数据的大小
- MC7：以字节为单位的 MC7 代码的大小、或 DB 用户数据的大小
- 可编程控制器的装入存储器的大小
- 可编程控制器的工作存储器的大小：仅在识别出硬件分配时显示。

出于显示的目的，它将与块是位于在线视图的窗口中还是位于离线视图的窗口中无关。

SIMATIC 管理器中的显示(详细视图)

如果块文件夹已打开，且选择了“详细视图”，则无论块文件夹是位于在线视图还是位于离线视图的窗口中，工作存储器的要求都将显示在项目窗口中。

通过选择所有相关的块，您可计算出块长度的总和。此时，所选块的总和将显示在SIMATIC管理器的状态栏中。

对于不能下载到可编程控制器的块，将不显示任何长度(例如变量表)。

将不在详细视图中显示可编程设备(PG/PC)上的块长度。

9.3.5 比较块

介绍

要比较处于不同位置的块，可从下列方法中任选一种来启动块的比较过程：

- 转到 SIMATIC 管理器并选择**选项 > 比较块**菜单命令。在所显示的“比较块 - 结果”对话框中，单击“跳转到”按钮。比较结果将显示在程序编辑器(LAD/FBD/STL)中的“比较”标签中。
- 跳转到程序编辑器。选择**选项 > 比较在线/离线伙伴**菜单命令。

下面的章节将介绍块比较过程如何进行的。在下面的讨论中，将保持逻辑块(OB、FB、FC)和数据块(DB)之间的区别。

块比较如何进行：逻辑块

在过程的第一步，STEP 7 将对需要比较的逻辑块接口的时间标志进行比较。如果这些时间标志完全相同，则 STEP 7 认为其接口相同。

如果时间标志不一样，那么，STEP 7 随后将以段为单位逐步地比较接口的数据类型。当发现差别时，STEP 7 将确定段中的第一个差异；也就是说，在所有情况下，都是各自声明范围中的**第一个**差异。在比较中也包括了多事例与 UDT。如果段中的数据类型相同，则 STEP 7 将接下来比较变量的初始值。所有的差异均将显示。

在第二步中，STEP 7 将逐个程序段地检查代码(倘若没有选择“执行代码比较”选项，但单击程序编辑器中的“跳转到”按钮时，将仍然对代码进行比较)。

首先，检测插入或删除的程序段。比较的结果中将显示只在一个块中出现的程序段。这些程序段将具有注释“仅存在于”。

随后，将对其余的程序段进行比较，直到找到语句中的第一个差异。将按下列方式对语句进行比较：

- 对于设置“绝对地址具有优先权”，将以绝对地址为基础
- 对于设置“符号具有优先权”，将以符号为基础

如果语句的操作符和地址是一样的，则认为语句是完全相同的。

如果将要比较的块使用了不同的程序设计语言进行编程，那么，STEP 7 将执行以 STL 语言为基础的比较。

离线-离线比较的特性：

与离线-在线的比较不同，在离线-离线的比较中，STEP 7 还将检测是否出现了不同的变量名称。这一附加步骤不适用于离线-离线的比较，因为只有替换符号可供在线使用。

块程序段和程序行的注释以及其他块属性(例如 S7-PDIAG 信息和消息)均不作比较。

块比较如何进行：数据块

在过程的第一步，STEP 7 将对需要比较的数据块接口的时间标志进行比较(同逻辑块)。如果这些时间标志完全相同，则 STEP 7 认为数据结构相同。

如果接口时间标志不同，则STEP 7将接下来比较数据结构，直到找到第一个差异。如果位于段中的数据结构完全相同，则STEP 7随后将比较初始值和当前值。所有的差异均将显示。

离线-离线比较的特性：

与离线-在线的比较不同，在离线-离线的比较中，STEP 7 还将检测是否出现了不同的变量名称。这一附加步骤不适用于离线-离线的比较，因为只有替换符号可供在线使用。

数据块中使用的 UDT 的注释和结构将不作比较。

块比较如何进行：数据类型(UDT)

在过程的第一步，STEP 7 将对需要比较的数据类型接口的时间标志进行比较(同数据块)。如果这些时间标志完全相同，则 STEP 7 认为数据结构相同。

如果接口时间标志不同，则STEP 7将接下来比较数据结构，直到找到第一个差异。如果位于段中的数据结构完全相同，则STEP 7随后将比较初始值。所有的差异均将显示。

块比较如何进行：程序编辑器中的比较

1. 打开将要与已下载版本进行比较的块。
2. 选择选项 > 比较在线/离线伙伴菜单命令。
   如果在线伙伴是可访问的，那么，比较的结果随后将显示在程序编辑器下部的“7：比较”标签中。
   提示：如果两个网络已确定是“不同的”，那么，您只需通过双击相关网络的行即可将其打开。

块比较如何进行：SIMATIC 管理器中的比较

1. 在SIMATIC管理器中,选择要比较的块文件夹或块。
2. 选择**选项 > 比较块**菜单命令。
3. 在所显示的“比较块”对话框中，选择比较的类型(在线/离线或路径1/路径2)。
4. 对于路径1/路径2比较：在SIMATIC管理器中,选择要比较的块文件夹或块。这些块随后均将自动输入到对话框中。
5. 如果还希望比较SDB，则可选择“包括SDB”复选框。
6. 如果也希望比较代码，则可选择“执行代码比较”复选框。在详细的比较中，除了块的与执行相关的部分(接口和代码)以外，用于局部变量和参数的名称中的任何改变均将显示。此外，您可选择“包括以不同编程语言创建的块”复选框来比较使用不同编程语言(例如AWL、FUP....)创建的块。此时，将基于STL对块进行比较。
7. 通过单击“确定”确认对话框中的设置。

   比较的结果将显示在“比较块 - 结果”对话框中。
8. 为了显示已比较块的属性(即上一次修改的时间、检验和等等)，可单击该对话框中的“详细资料”按钮。
   单击“跳转到”按钮，可打开程序编辑器，比较的结果将显示在它的下层窗口中。

注意

当比较离线和在线的块文件夹时，将只比较可装载的块类型(OB、FB...)。

当比较离线/在线或路径1/路径2时，将比较多重选择中所包含的全部块均，即使它们中有一些并不是可装载的(例如变量表或UDT)。

9.3.6 重新连接

下列块和地址均可重新进行布线：

- 输入、输出
- 存储器位、定时器、计数器
- 功能、功能块

如要重新布线：

1. 选择“块”文件夹，该文件夹包含有在SIMATIC管理器中希望重新接线的各个块。
2. 选择菜单命令**选项 > 重新接线**。
3. 在“重新布线”对话框中的表格中输入所需要的替代值(旧的地址/新的地址)。
4. 如果希望对地址区重新布线，可选择选项“在指定地址区内的所有地址”(BYTE、WORD、DWORD)。
   实例：实例:输入IW0和IW4作为地址区。于是将把地址I0.0 – I1.7重新布线为地址I4.0 – I5.7。于是就不需要将已重新布线的地址区中的地址(例如，I0.1)再单独输入到表格中。
5. 点击“确定”按钮。

这将启动重新布线过程。在重新布线完成之后，您可在对话框中指定您是否希望阅读关于重新布线的信息文件。该信息文件包含有地址列表“旧的地址”和“新的地址”。并按布线过程每次执行时的编号列出各个块。

在进行重新布线时，应注意以下方面：

- 在重新布线(也就是重新命名)一个块时，新的块在当前可能不存在。如果块存在，将中断该过程。
- 在重新布线一个功能块(FB)时，背景数据块将自动分配给已重新布线的 FB。但背景数据块将不发生变化，也就是说，DB 编号将保留。

9.3.7 块和参数的属性

在关于系统属性的参考帮助中可找到关于属性的描述：

跳转到关于块、系统属性的语言描述与帮助

9.4 使用库进行工作

库用于存储 SIMATIC S7/M7 中可重复使用的程序组件。既可以从现有的项目中将程序组件复制到库中，也可以在与其它项目无关的库中直接创建。

如果将希望在 S7 程序的库中多次使用的块存储下来，则可节省大量的编程时间和工作量。可将其从此处复制到需要的用户程序中。

为创建库中的 S7/M7 程序，同样的功能也适用于项目 – 除了调试功能以外。

创建库

与项目完全一样，您也可使用菜单命令**文件 > 新建**来创建库。新的库将创建在您选择菜单命令**选项 > 自定义**时在“常规”标签中为库所设置的目录中。

注意

SIMATIC管理器允许使用长度多于8个字符的名称。然而，库目录的名称将截取到8个字符。因此，库名称必须在头8个字符之内有所区分。名称不区分大小写。当在浏览器中打开该目录时，将再次显示完整的名称，但在对目录进行浏览时，将只出现已截短的名称。

请注意，您不能在旧的STEP 7版本项目中使用新STEP 7版本的库中的块。

打开库

如要打开现有的库，可输入菜单命令**文件 > 打开**。然后在随后出现的对话框中选择一个库。于是打开了库窗口。

注意

如果在库列表中无法找到您所需要的库，则可单击“打开”对话框中的“浏览”按钮。于是标准Windows浏览器将显示一个目录结构，可在其中对库进行搜索。

请注意，文件的名称总是对应于库创建时的原有名称，这意味着在SIMATIC 管理器中进行的任何名称修改在文件这一级都不能进行。

当您选择一个库时，它将被添加到库列表中。您可使用菜单命令**文件 > 管理**对库列表中的条目进行修改。

复制库

通过使用菜单命令**文件 > 另存为**，以另一个名称保存库，从而复制了这个库。

使用菜单命令**编辑 > 复制**，可对库的某一部分例如程序、块、源文件等进行复制。

删除库

使用菜单命令**文件 > 删除**，可删除一个库。

使用菜单命令**编辑 > 删除**，可删除库的一部分，例如程序、块、源文件等。

9.4.1 库的层次结构

正如项目一样，可按分层的方式对库进行结构化：

- 库包含 S7/M7 程序。
- S7 程序可包含一个“块”文件夹(用户程序)、一个“源文件”文件夹、一个“图表”文件夹、以及一个“符号”对象(符号表)。
- M7 程序可包含用于可编程 M7 模块的图表与 C 程序，以及用于数据块和变量表的“符号”对象(符号表)与“块”文件夹。
- “块”文件夹包含可下载到 S7 CPU 中的块。不会将文件夹中的变量表(VAT)和用户自定义数据类型下载到 CPU 中。
- “源文件”文件夹包含使用各种不同编程语言所创建的程序的源文件。
- “图表”文件夹包含 CFC 图表(仅适用于安装了 S7 CFC 选项包时)。

当插入一个新的 S7/M7 程序时，将在其中自动插入“块”文件夹、“源文件”文件夹(仅适用于 S7)、以及“符号”对象。

9.4.2 标准库概述

STEP 7 标准软件包包含有下列标准库

- **系统功能块**：系统功能块(SFB)和系统功能(SFC)
- **S5-S7转换块**：转换STEP 5程序的块
- **IEC功能块**：用于IEC功能的块，例如，用于处理时间和日期信息、比较操作、字符串处理以及选择最小/最大值。
- **组织块**：缺省组织块(OB)
- **PID控制块**：用于PID控制的功能块(FB)
- **通讯块**：用于SIMATIC NET CP的功能(FC)与功能块(FB)。

- **TI-S7转换块**：一般用途的标准功能
- **其它块**：用于时间标记以及用于TOD同步的块

在安装选项软件包时，可能要添加其它的块。

删除和安装所提供的块

可在 SIMATIC 管理器中删除所提供的块，然后再重新安装它们。运行 STEP 7 安装程序，以便重新安装库。

注意

在安装STEP 7时，将总是复制所提供的库。如果编辑这些库，那么，当STEP 7再次重新安装时，所修改过的库将被原有库所覆盖。

为此，在进行任何修改之前，都应先复制所提供的库，然后只需对备份进行编辑即可。

---

10 创建逻辑块

10.1 创建逻辑块的基本步骤

10.1.1 程序编辑器窗口的结构

程序编辑器的窗口可拆分为下列区域：

表格

“程序元素”标签将显示一个程序元素表格，其中的程序元素均可插入到 LAD、FBD 或 STL 程序中。“调用结构”标签表示当前 S7 程序中的块的调用层次。

变量声明

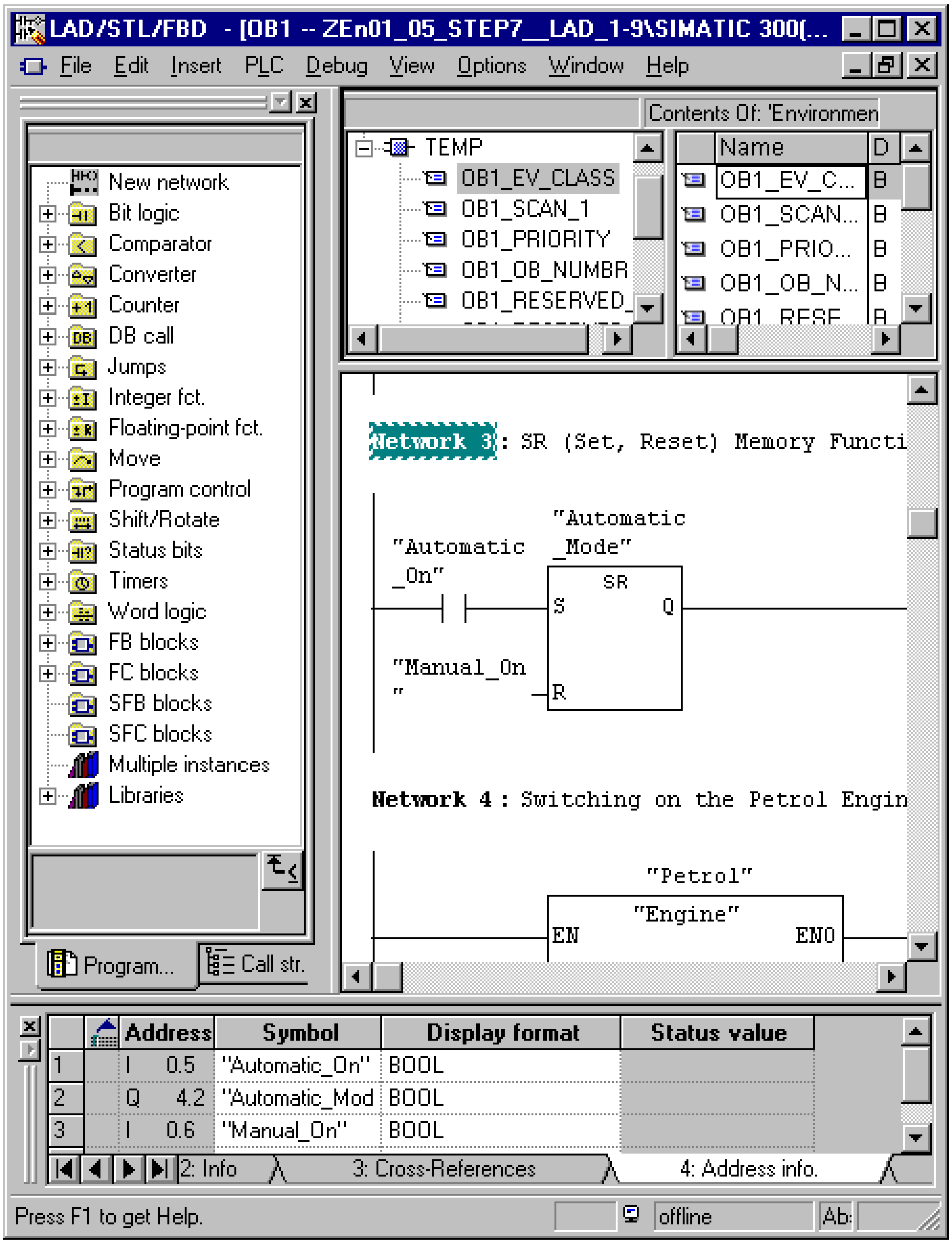
变量声明分为“变量表”和“变量详细视图”部分。

指令

指令表显示了将由 PLC 进行处理的块代码。它由一个或多个程序段组成。

详细信息

“详细情况”窗口中的各种不同标签提供了众多的功能，例如，用于显示出错消息、对符号进行编辑、生成地址信息、对地址进行控制、对块进行比较的功能以及对硬件诊断时的出错定义进行编辑的功能。



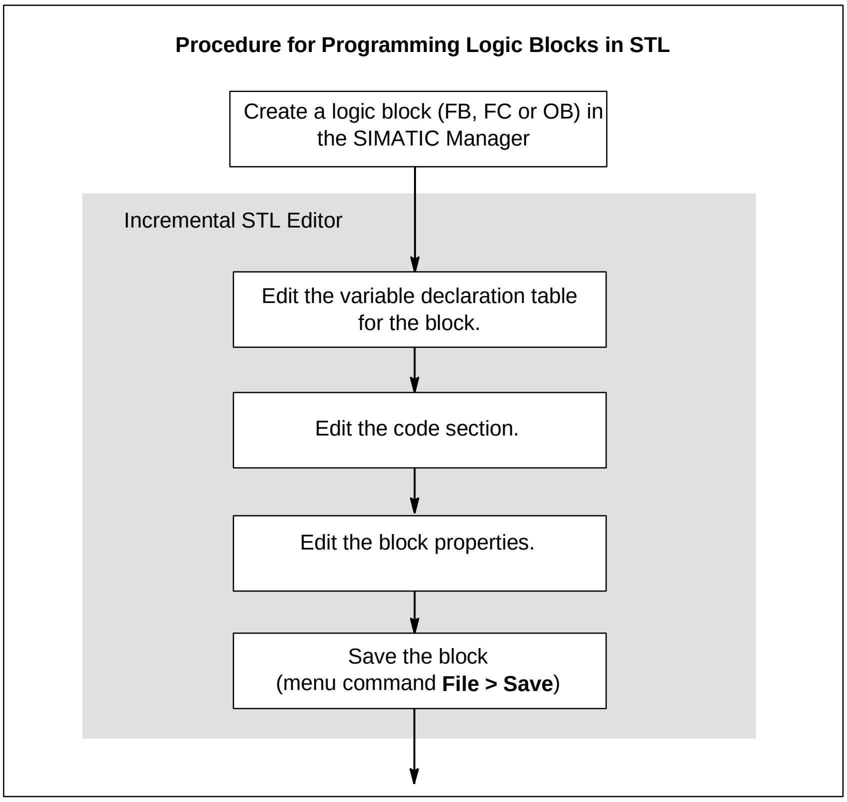
LAD/STL/FBD - [OB1 -- ZEn01_05_STEP7__LAD_1-9\SIMATIC 300(...
File Edit Insert PLC Debug View Options Window Help
New network
Bit logic
Comparator
Converter
Counter
DB call
Jumps
Integer fct.
Floating-point fct.
Move
Program control
Shift/Rotate
Status bits
Timers
Word logic
FB blocks
FC blocks
SFB blocks
SFC blocks
Multiple instances
Libraries
Program...
Call str.
TEMP
OB1_EV_CLASS
OB1_SCAN_1
OB1_PRIORITY
OB1_OB_NUMBR
OB1_RESERVED_
Contents Of: 'Environmen
Name
D
OB1_EV_C...
OB1_SCAN...
OB1_PRIO...
OB1_OB_N...
Network 3: SR (Set, Reset) Memory Functi
"Automatic _Mode"
"Automatic _On"
SR
S
Q
"Manual_On"
R
Network 4 : Switching on the Petrol Engin
"Petrol"
"Engine"
EN
ENO
Address
Symbol
Display format
Status value
1 I 0.5 "Automatic_On" BOOL
2 Q 4.2 "Automatic_Mod BOOL
3 I 0.6 "Manual_On" BOOL
2: Info
3: Cross-References
4: Address info.
Press F1 to get Help.
offline
Ab:

10.1.2 创建逻辑块的基本步骤

逻辑块(OB、FB、FC)由变量声明段、代码段及其属性等组成。在编程时，您必须编辑下列三个部分：

- **变量声明**：在变量声明中，您可指定参数、参数的系统属性以及块专用局部变量。
- **代码段**：在代码段中，您可对将要由可编程控制器进行处理的块代码进行编程。它由一个或多个程序段组成。要创建程序段，可使用各种编程语言，例如，梯形逻辑图(LAD)、功能块图(FBD)、或语句表(STL)。
- **块属性**：块属性包含了其它附加的信息，例如由系统输入的时间标志或路径。此外，您也可输入自己的详细资料，例如名称、系列、版本以及作者，还可为这些块分配系统属性。

原则上，您编辑逻辑块各部分的次序并不重要。当然，您也可对其进行改正和对其进行添加。



注意

如果要利用符号表中的符号，您应首先检查它们是否完整并进行必要的修正。

10.1.3 LAD/STL/FBD程序编辑器的缺省设置

在开始进行编程之前，应先熟悉编辑器中的设置，以便使编程更容易、更顺利。

使用菜单命令**选项 > 自定义**可打开用标签细分的对话框。在各种不同的标签中，为对块进行编程，可进行如下缺省设置，例如，在“常规”标签中：

- 用于文本和表格的字体(字型和字号)。
- 对于新块，是否希望显示符号和注释。

使用**视图 >…**菜单中的命令可在编辑期间修改语言、注释、以及符号等设置。

例如，可以改变用于高亮显示“LAD/FBD”标签中的程序段或语句行的颜色。

10.1.4 块和源文件的访问权限

在编辑项目时，经常要使用公共的数据库，这意味着在同一时间内，可能有许多人想访问同一个块或数据源。

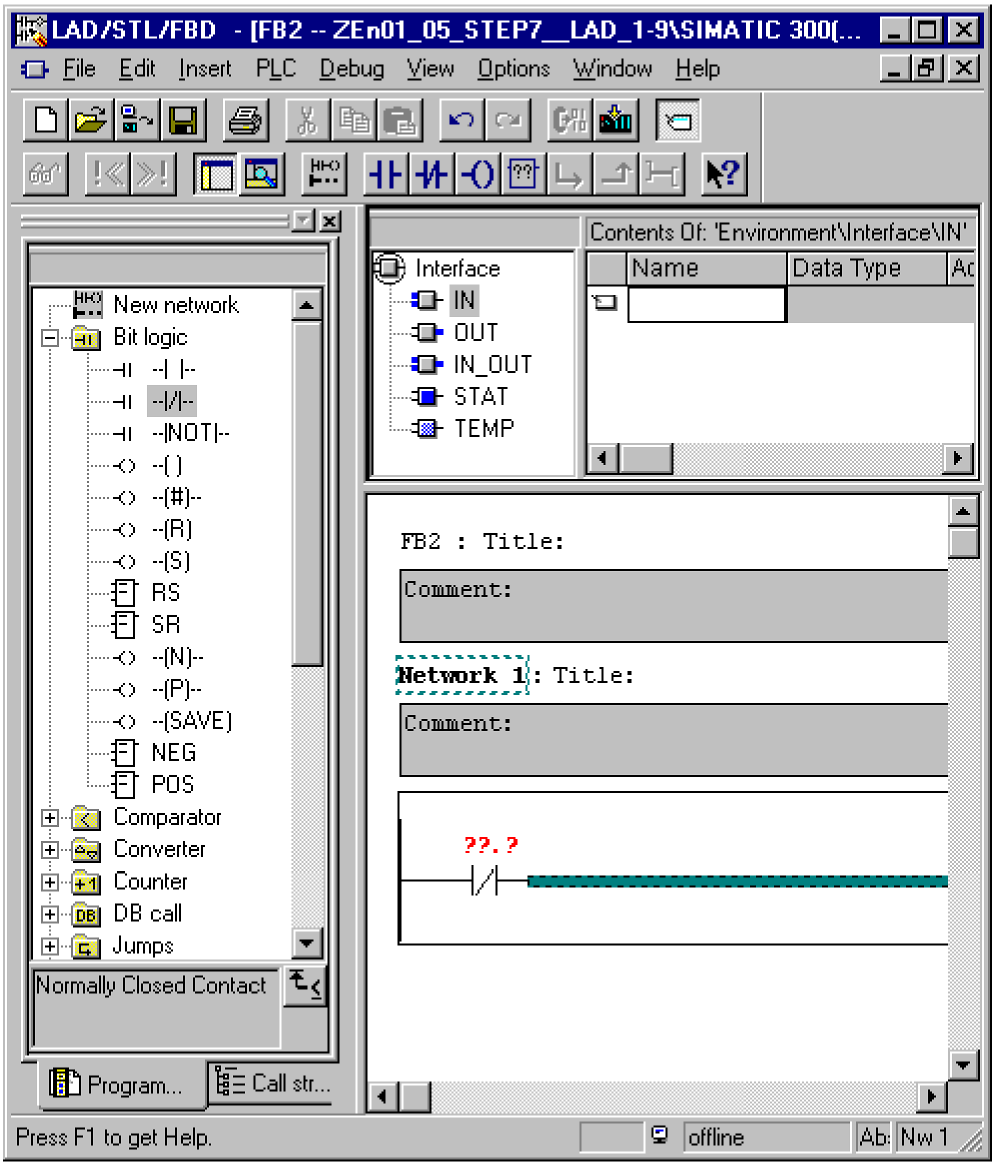
如下分配读/写访问权限：

- 离线编辑：
  当您试图打开一个块/源文件时，将对此进行检查，以确定您是否具有该对象的“写”访问权限。如果块/源文件已打开，则只能使用备份进行工作。如果随后试图保存备份，则系统将询问是否希望覆盖原来的块或文件，或者以新的名称保存备份。
- 在线编辑：
  当您通过所组态的连接打开一个在线块时，将关闭相应的离线块，以避免同时对其进行编辑。

10.1.5 程序元素表中的指令

概述窗口中的“程序元素”标签提供了LAD和FBD元素以及已声明的多重背景、预组态的块和来自库中的块。通过菜单命令**视图 > 表格**可对标签进行访问。使用菜单命令**插入 > 程序元素**也可将程序元素插入到代码段中。

LAD 中“程序元素”标签的实例

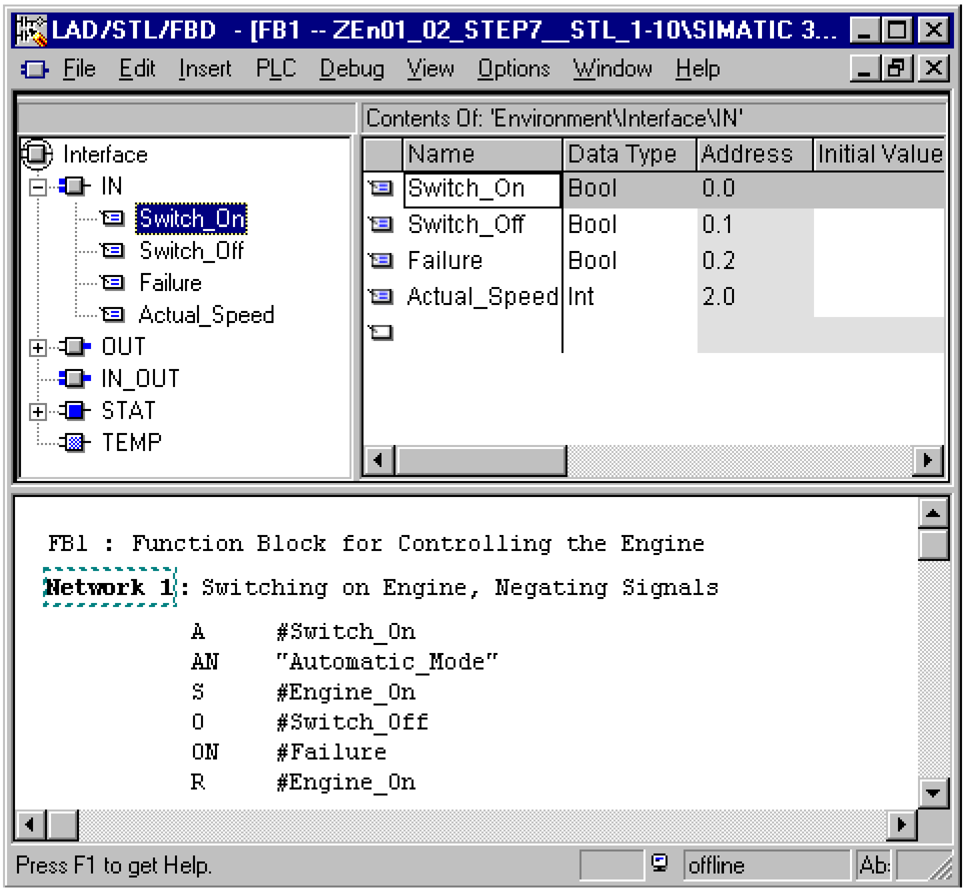


10.2 编辑变量声明

10.2.1 在逻辑块中使用变量声明

在打开一个逻辑块之后，所打开的窗口上半部分将包括块的变量表和变量详细视图，而窗口下半部分将包括将在其中对实际的块代码进行编辑的指令表。

实例：STL中的变量视图与指令表



在变量详细视图中，可指定块的局部变量和形式参数以及参数的系统属性。这将具有下列功能：

- 在变量声明期间，将在本地数据堆栈中为临时变量保留足够的存储空间，而对于功能块而言，则要为以后将要链接的背景数据块中的静态变量保留足够的存储空间。
- 在设置输入、输出、以及输入/输出参数时，也可在程序中为块的调用指定“接口”。
- 当在功能块中声明变量时，这些变量(除了临时变量以外)也将决定与功能块相联结的每个背景数据块的数据结构。
- 通过设置系统属性，例如，可为消息与连接函数的组态、操作员控制与监视功能以及过程控制组态等分配特定的属性。

10.2.2 变量详细视图与指令表之间的联系

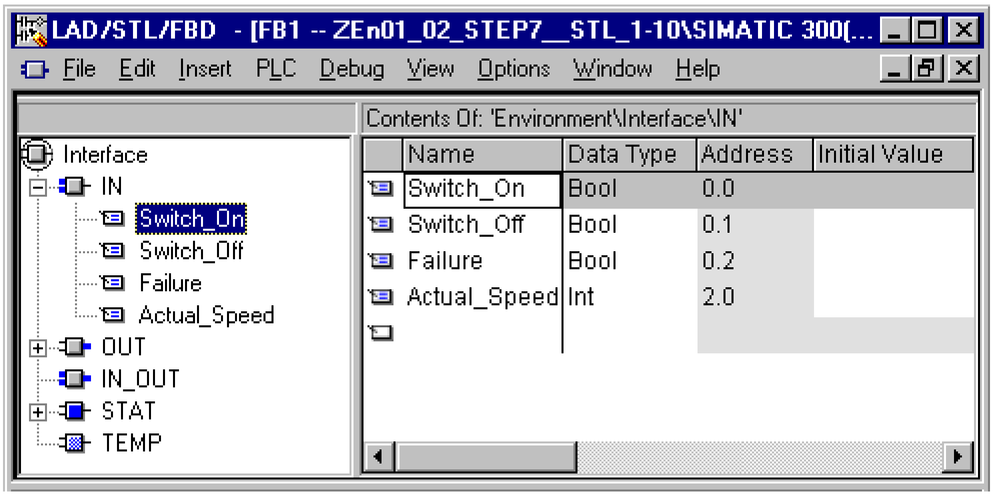
逻辑块的变量声明与指令表是紧密关联的，因为在变量声明中为进行编程而指定的名称也将用于指令报表中。因此，在变量声明中所做的所有修改都将影响整个指令表。

变量声明中的动作	代码段的响应
正确的新输入	如果出现无效的代码，则先前尚未说明的变量现在将变为有效。
类型不变，正确的名称改变	符号将以其新名称立即显示在每一个地方
将正确的名称变为无效的名称	代码保持不变
将无效的名称变为正确的名称	如果出现无效的代码，则它将变为有效
类型变化	如果出现无效的代码，则它将变为有效，但如果出现有效的代码，则这可能变为无效
删除代码中所使用的变量(符号名称)	有效代码将变为无效

对注释的修改、新变量的错误输入、初始值的更改、或删除未使用的变量等均不对指令表产生任何影响。

10.2.3 变量声明窗口的结构

变量声明窗口是由变量和变量详细视图的所组成的。



在已经生成并打开新的代码块之后，将显示一个缺省的变量表。它将只列出所选块允许的声明类型(in、out、in_out、stat、temp)，就是按照规定的次序。在生成新的 OB 之后，您可编辑所显示的缺省变量声明。

各种不同块类型的本地数据的允许数据类型，请参见将数据类型分配给代码块的本地数据。

10.3 变量声明中的多实例

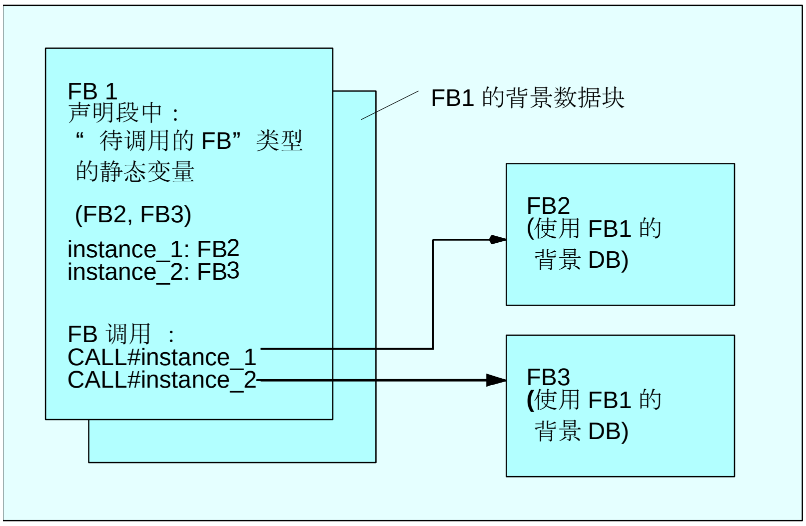
10.3.1 使用多重背景

可能由于正在使用的 S7 CPU 的性能原因(例如，存储器的容量)，想要或不得不使用数量受到限制的背景数据的数据块。如果在用户程序的 FB 中要调用现有的其它功能块(FB 的调用体系)，那么，您可调用其余的这些没有自己(附加的)背景数据块的功能块。

使用如下解决方法：

- 将希望作为静态变量调用的功能块包含到正在调用的功能块的变量说明中。
- 在该功能块中，调用没有自己(附加的)背景数据块的其它功能块。
- 这将把背景数据集中在一个背景数据块中，这意味着您可更有效地使用这些现有的数据块。

下面的实例说明了所描述的解决方法：FB2 和 FB3 均使用了从中对其进行调用的功能块 FB1 的背景数据块。



唯一的要求：您必须“告诉”正在调用的功能块，您正在调用哪一个背景以及这些背景都是什么(FB)类型。这些具体的细节都必须输入到正在调用的功能块的声明窗口中。所使用的功能块必须至少具有来自数据区的一个变量或参数(不能使用VAR_TEMP)。

如果在CPU运行时希望进行在线修改，则切勿使用多背景数据块。当使用背景数据块时，必须保证重新装载对系统没有大的影响。

10.3.2 多重背景的声明规则

下列规则均适用于多重背景的声明：

- 只有在使用版本 2 以上的 STEP 7 所创建的功能块中才可能对多重背景进行声明(参见功能块属性中的块属性)。
- 为了声明多重背景，必须将功能块创建为具有多重背景能力的功能块(STEP 7版本x.x的缺省设置；可在编辑器中使用**选项 > 自定义**取消激活)。
- 必须将背景数据块分配给在其中对多重背景进行声明的功能块。
- 只能将多重背景声明为静态变量(说明类型为“stat”)。

注意

- 您也可为系统功能块创建多重背景。
- 如果没有创建能够具有多重背景功能的功能块，而您又希望该功能块具有这种属性，那么，您可根据功能块生成一个源文件，随后删除其中的块属性CODE_VERSION1，然后重新对功能块进行编译即可。

10.3.3 在变量声明窗口中输入一个多重背景

1. 打开将要从中调用下一级功能块的功能块。
2. 在调用功能块的变量声明中，为每一次功能块调用定义一个静态变量，用于保存那些不希望为其创建背景数据块的背景数据。
   - 在变量表中，选择变量类型“STAT”。
   - 在变量详细视图的“名称”列中为 FB 调用输入一个名称
   - 在“数据类型”列中输入想要作为绝对地址而调用的功能块，或具有其符号名称的功能块。
   - 在注释列中，可输入所需要的任何解释。

代码段中的调用

当声明完毕多重背景时，即可使用 FB 调用，而无需指定一个背景数据块。

实例：如果静态变量“名称：Motor_1、数据类型：FB20”已定义，则可如下调用背景：

Call Motor_1　　　// 调用没有背景数据块的 FB20

10.4 关于输入语句和注释的常规注意事项

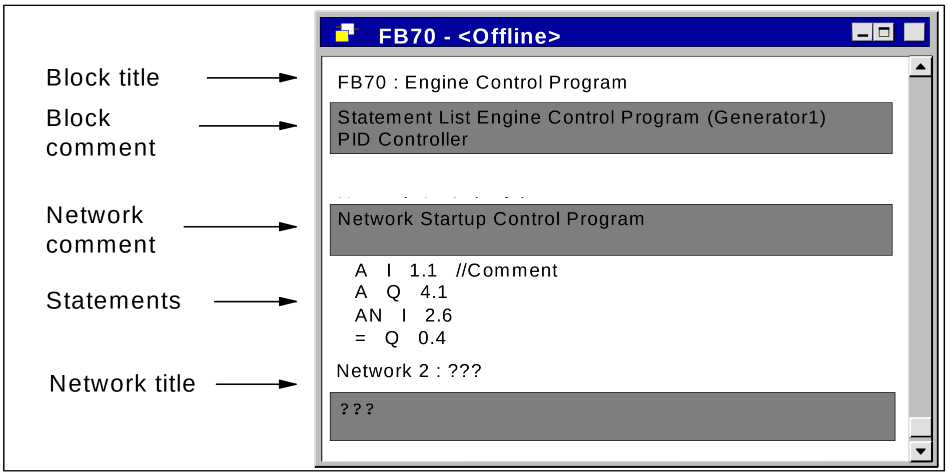
10.4.1 代码段的结构

在代码段中，根据所选择的编程语言，可通过在程序段中输入合适的语句来对逻辑块的次序进行编程。在输入一条语句之后，编辑器将立即执行语法检查，并使用红色和斜体显示所有的错误。

逻辑块的代码段通常包含许多程序段，这些程序段则由语句列表组成。

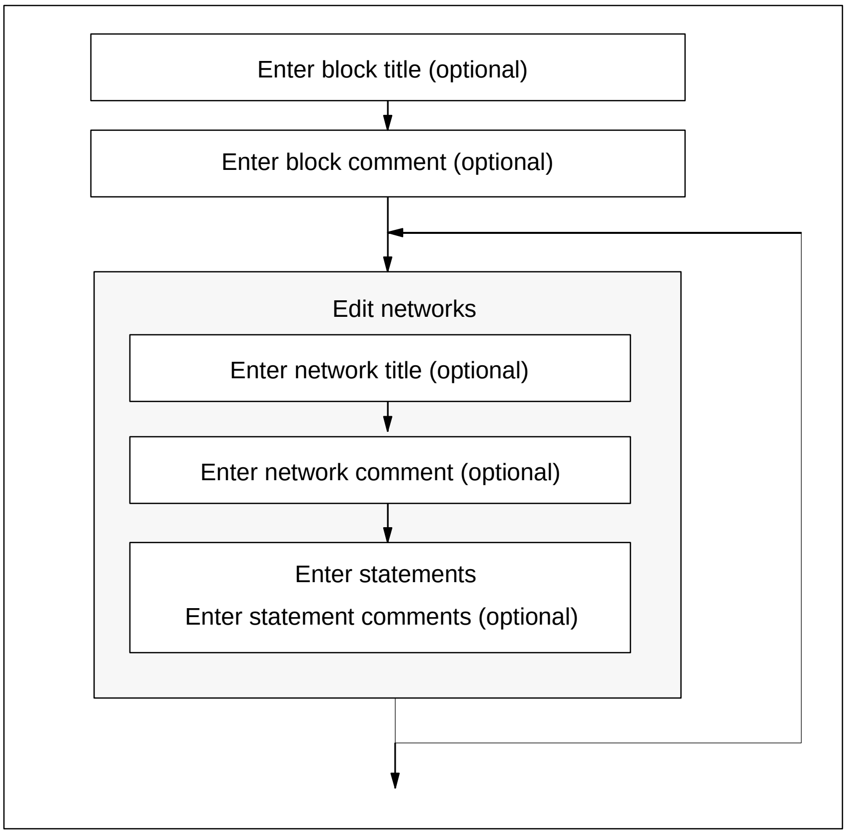
在代码段中，可编辑块标题、块注释、程序段标题、程序段注释、以及程序段内的语句行。

使用 STL 编程语言的代码段的结构实例



10.4.2 语句的输入步骤

可按任意次序对部分代码段进行编辑。当首次编写块时，建议按如下的步骤进行操作：



既可在改写模式下，也可在插入模式下进行修改。使用INSERT键可在这两种模式之间切换。

10.4.3 在程序中输入共享符号

使用菜单命令**插入 > 符号**，可将符号插入到程序的代码段中。如果光标位于字符串的开头、结尾或中间，那么，就已经选择了一个以该字符串开头的符号 - 如果这样的符号存在的话。如果改变字符串，则进行的选择也将在列表中更新。

用于字符串开头和结尾的分隔符可以是：空格、句点、冒号。在共享符号内将不解释任何分隔符。

为输入符号，可按如下操作进行：

1. 输入程序中所需符号的第一个字母。
2. 同时按下CTRL和J键，以显示符号列表。以输入的字母开头的第一个符号已经选中。
3. 按下回车键输入符号或选择另外的符号。

随后即可输入加引号的符号，以替换输入的第一个字母。

通常会发生下列情况：如果光标位于字符串的开头、结尾、或中间，当插入符号时，该字符串将被用引号括起来的符号所替换。

10.4.4 块和程序段的标题与注释

注释将使您的用户程序更易于阅读，从而使调试和查找错误更容易，也更有效。它们是程序文档的一个重要组成部分，毫无疑问应加以充分利用。

LAD、FBD 和 STL 程序中的注释

可供使用的注释如下：

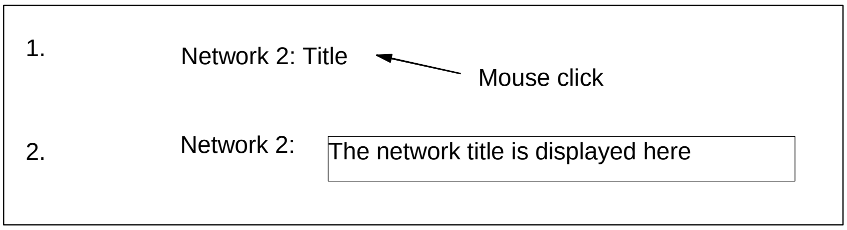
- 块标题：用于块的标题(最多 64 个字符)
- 块注释：对整个逻辑块进行记录，例如，块的用途
- 程序段标题：用于程序段的标题(最多 64 个字符)
- 程序段注释：对单个程序段的功能进行记录
- 变量详细视图中的注释列：为所声明的本地数据加上注释
- 符号注释：当在符号表中定义地址的符号名称时所输入的关于地址的注释。使用菜单命令**视图 > 显示 > 符号信息**可显示这些注释。

在逻辑块的代码段中，可输入块标题和程序段标题，以及块注释或程序段注释。

块标题或程序段标题

为输入块标题或程序段标题，可将光标放置在块名称或程序段名称右边的单词“标题”上(例如，程序段1：标题：)单击。即可打开一个供您在其中输入标题的文本框。其长度最多可达到64个字符。

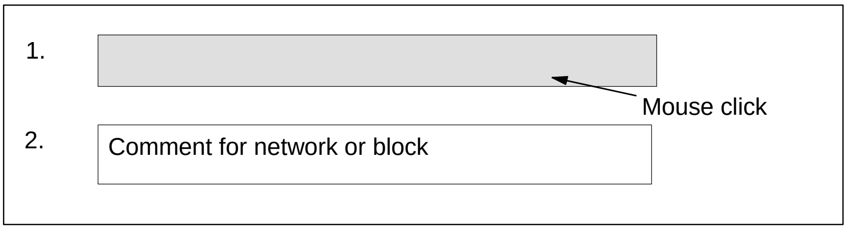
块注释属于整个逻辑块。这里，它们可对块的功能进行注释。程序段注释属于单独的程序段以及与程序段有关的文档细节。



为自动分配程序段标题，可选择菜单命令**选项 > 设置**并单击“常规”标签页中的选项“自动分配程序段标题”。所输入第一个地址的符号注释随后将作为程序段标题应用。

块注释与程序段注释

使用菜单命令**视图 > 显示 > 注释**可打开和关闭灰色注释域的视图。双击注释域可打开您现在可在其中输入解释的文本框。对于块注释和程序段注释，允许您每个块使用64 K字节。



10.4.5 输入块注释与程序段注释

1. 使用菜单命令**视图 > 显示 > 注释**激活注释(在菜单命令的前面可见到一个复选标记)。
2. 通过使用鼠标单击，将光标放置在块名称下的或程序段名称下的灰色域中。灰色注释域将出现白色并具有一个边框。
3. 在打开的文本框中输入您的注释。对于块注释和程序段注释，允许您每个块使用64 K字节。
4. 通过鼠标单击文本框的外面、按下TAB键、或使用组合键SHIFT+TAB都可退出文本框。
5. 如果您再次选择菜单命令**视图 > 显示 > 注释**，那么，您可重新关闭注释(复选标记将消失)。

10.4.6 使用网络模板进行工作

编程块时，如果您愿意多次使用网络，可将这些网络作为网络模板存储在库中，适当时可使用通配符(例如，地址)。在您创建网络模板之前，库必须可供使用。

创建网络模板

如果必要，可在SIMATIC管理器中创建一个新库。选择菜单命令**插入 > 程序 > S7程序**，可将程序插入到库中。

1. 打开您希望通过其中所含网络来创建网络模板的块。
2. 在已打开的块中，根据需要，使用通配符替换标题、注释或地址。您可使用字符串%00至%99作为通配符。地址的通配符均显示为红色。此处，这不成问题，因为在您创建网络模板之后您将不保存块。日后，当您将网络模板插入到块中时，可使用合适的地址来替换通配符。
3. 选择您要在网络模板中包含的网络的“网络<编号>”。
4. 选择菜单命令编辑 > 创建网络模板。
5. 在所显示的对话框中为所用的各个通配符输入具有一定含义的注释。
6. 点击“确定”按钮。
7. 在所出现的浏览器中，选择网络模板库中的S7程序的**源文件文件夹**，然后为网络模板输入一个名称。
8. 点击“确定”按钮，确认输入。网络模板存储在所选择的库中。
9. 关闭块，不进行保存。

将网络模板插入到程序中

1. 打开您希望插入新网络的块。
2. 在所打开的块中，点击您希望在其后插入以网络模板为基础的新网络的网络。
3. 打开“程序元素”标签(菜单命令**插入 > 程序元素**)。
4. 打开目录中相关库的“S7程序”文件夹。
5. 双击程序段模板。
6. 在对话框中，输入所需要的网络模板通配符的替换值。
7. 点击“确定”按钮。随后，网络模板插入到当前网络中。

注意

您也可将模板从标签中拖放到编辑器窗口中。

10.4.7 用于代码段错误的搜索功能

代码段中的错误很容易通过其红色标记进行识别。为了使屏幕上的可见区域以外的错误浏览起来更容易，编辑器提供了两种搜索功能**编辑 > 跳转到 > 前一个错误/下一个错误**。

对错误的搜索将不限于一个程序段。这意味着将搜索整个代码段，而不仅仅是一个程序段或当前在屏幕上可见的区域。

如果您使用菜单命令**视图 > 状态栏** 激活状态栏，则在此显示与所找到的错误相关的注释。

您也可在改写模式下纠正错误和进行修改。使用 INSERT 键可在插入模式和改写模式之间进行切换。

10.5 编辑代码段中的LAD单元

10.5.1 梯形图编程的设置

设置梯形图布局

您可按照梯形图表示类型设置创建程序时的布局。您选择的格式(A4纵向/横向/最大尺寸)将影响一个梯级中所能显示的梯形图元素的数量。

1. 选择菜单命令**选项 > 自定义**。
2. 在下面的对话框中选择“LAD/FBD”标签。
3. 从“布局”列表框中选择所需要的格式。输入所需要的格式尺寸。

打印设置

如果想要打印输出梯形图代码段，那么，在开始对代码段进行编程之前，应设置合适的页面格式。

“LAD/FBD”标签中的设置

在使用菜单命令**选项 > 自定义**对其进行访问的“LAD/FBD”标签中，可进行一些基本的设置，例如，与布局和地址域宽度有关的设置。

10.5.2 梯形图元素的输入规则

在“用于 S7-300/400 对块进行编程的梯形图”手册中或梯形图在线帮助中，可以找到关于梯形图编程语言表示的描述。

一个梯形图程序段可由多个分支中的许多元素组成。所有的元素和分支必须进行连接；左电源线不算作连接(IEC 1131-3)。

当在梯形图中进行编程时必须遵循一些原则。出错消息将告诉您产生的错误。

关闭梯形图程序段

每个梯形图程序段都必须使用线圈或逻辑方框来关闭。不能使用下列梯形图元素来关闭程序段：

- 比较框
- 中间变量输出_/(#)_/
- 用于上升沿_/(P)_/或下降沿_/(N)_/计算的线圈

定位框

用于逻辑框连接的分支起始点必须始终为左电源线。逻辑操作或其它逻辑框可出现在逻辑框前面的分支中。

定位线圈

线圈将自动定位在程序段的右边沿，它们在这里构成了分支的末端。

例外：用于中间变量输出_/(#)_/以及上升沿_/(P)_/或下降沿_/(N)_/计算的线圈均不能放置在分支的最左边，也不能放置在分支的最右边。它们二者在并行分支中均不允许。

某些线圈需要布尔型逻辑操作，而另外的一些线圈则不一定需要布尔型逻辑操作。

- 需要布尔型逻辑操作的线圈：
  - 输出_/( )、置位输出_/(S)、复位输出_/(R)
  - 中间变量输出_/(#)_/、上升沿_/(P)_/、下降沿_/(N)_/
  - 所有的计数器和定时器线圈
  - 如果为非(Not)，则跳转_/(JMPN)
  - 主控制继电器接通_/(MCR<)
  - 将 RLO 保存到 BR 存储器_/(SAVE)
  - 返回_/(RET)
- 不允许布尔型逻辑操作的线圈：
  - 主控制继电器激活_/(MCRA)

- 主控制继电器取消激活_/(MCRD)
- 打开数据块_/(OPN)
- 主控制继电器断开_/(MCR>)

其它所有线圈既可以带布尔型逻辑操作，也可以不带。

下列线圈不能用作并行输出：

- 如果为(Not)，则跳转_/(JMPN)
- 跳转_/(JMP)
- 来自线圈的调用_/(CALL)
- 返回_/(RET)

使能输入/使能输出

逻辑框的使能输入“EN”与使能输出“ENO”可进行连接，但这并非强制性要求。

删除与改写

如果一个分支仅由一个元素组成，则当删除了该元素时，整个分支也将删除。

当删除一个逻辑框时，与逻辑框的布尔型输入相连接的所有分支，除了主分支以外，都将删除。

改写模式可用来只改写同一类型的元素。

并行分支

- 从左到右画出 OR 分支。
- 并行分支向下打开，向上关闭。
- 并行分支将总是在所选梯形元素之后打开。
- 并行分支将总是在所选梯形元素后关闭。
- 为删除一个并行分支，可删除分支中的所有元素。当删除了分支中的最后一个元素时，分支于是被自动删除。

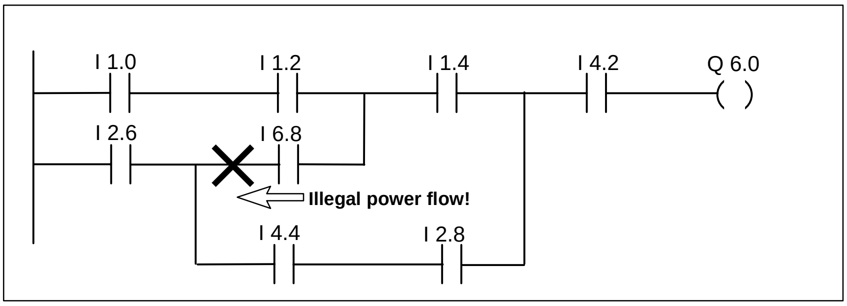
常量

不能给二进制链接分配常量(例如，真(TRUE)或假(FALSE))。取而代之，可使用 BOOL 数据类型的地址。

10.5.3 梯形图中的非法逻辑操作

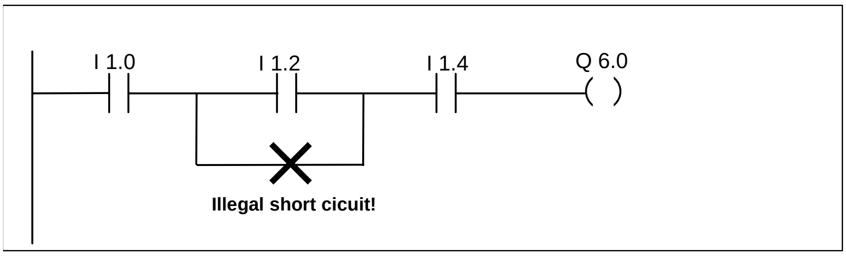
电流从右到左

不能创建可能导致电流反方向流动的任何分支。下图表示一个实例：当I1.4处的信号状态为“0”时，可能导致I6.8处产生从右到左的电流。这是不允许的。



短路

不能创建可能导致短路的任何分支。下图表示一个实例：



10.6 编辑代码段中的FBD单元

10.6.1 用于功能块图编程的设置

设置功能块图的布局

可以按照功能块图的表示类型来设置程序创建时的布局。您选择的格式(A4纵向/横向/最大尺寸)将影响一个梯级中所能显示的FBD元素的数量。

1. 选择菜单命令**选项 > 自定义**。
2. 在下面的对话框中选择“LAD/FBD”标签。
3. 从“布局”列表框中选择所需要的格式。输入所需要的格式尺寸。

打印设置

如果您想要打印输出 FBD 代码段，那么，在开始对代码段进行编程之前，应设置合适的页面格式。

“LAD/FBD”标签中的设置

在使用菜单命令**选项 > 自定义**对其进行访问的“LAD/FBD”标签中，可进行一些基本的设置，例如，与布局和地址域宽度有关的设置。

10.6.2 FBD元素的输入规则

在“用于 S7-300/400 – 编程块的功能块图”手册或 FBD 在线帮助中，可找到关于编程语言“FBD”的描述。

一个 FBD 程序段可由多个元素组成。所有的元素都必须互相连接(IEC 1131-3)。

当在 FBD 中编程时，必须遵循一些规则。出错消息将告诉您产生的错误。

输入并编辑地址和参数

当插入FBD元素时，字符??? 和...将用作地址和参数的代用字符。

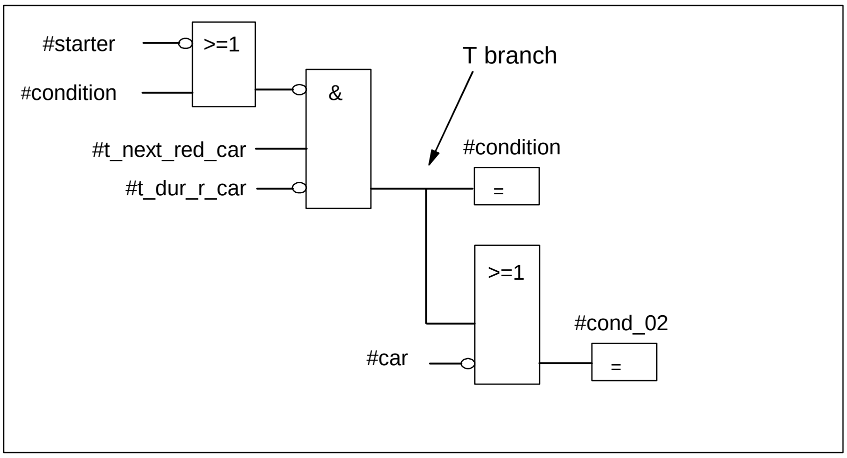
- 红色的字符???代表必须连接的地址和参数。
- 黑色的字符...代表可以连接的地址和参数。

如果将鼠标指针放置在代用字符上，则将显示所期望的数据类型。

定位框

可将标准的逻辑框(触发器、计数器、定时器、数学运算等)添加到具有二进制逻辑运算(&、>=1、XOR)的框中。该规则的例外情况是比较框：

在程序段中不能对带有单独输出的任何单独逻辑运算进行编程。然而，您可借助于分支对具有逻辑运算的字符串进行赋值。下图表示一个带有两个赋值操作的程序段。



下列逻辑框只能在逻辑字符串的右边沿处进行替换，在此，它们将关闭字符串：

- 设置计数器的值
- 分配参数并递增计数，分配参数并递减计数
- 分配脉冲定时器参数并启动，分配扩展脉冲定时器参数并启动
- 分配接通延迟/关闭延迟定时器参数并启动

某些逻辑框需要布尔型逻辑运算，而另外的一些逻辑框不一定需要布尔型逻辑运算。

需要布尔型逻辑运算的逻辑框：

- 输出、置位输出、复位输出_/[R]
- 中间变量输出_/[#]_/、上升沿_/[P]_/、下降沿_/[N]_/
- 所有的计数器和定时器框
- 如果为非(Not)，则跳转_/[JMPN]
- 主控制继电器接通_/[MCR<]
- 将 RLO 保存到 BR 存储器_/[SAVE]
- 返回_/[RET]

不允许进行布尔型逻辑运算的框：

- 主控制继电器激活[MCRA]
- 主控制继电器取消激活[MCRD]
- 打开数据块[OPN]
- 主控制继电器关闭[MCR>]

其它所有的框可以带布尔型逻辑运算，也可以不带。

使能输入/使能输出

可以连接逻辑框的使能输入“EN”与使能输出“ENO”，但这并非强制性要求。

删除与改写

当删除一个逻辑框时，与逻辑框的布尔型输入相连接的所有分支，除了主分支以外，都将删除。

改写模式可用来只改写同一类型的元素。

常量

不能将常量分配给二进制链接 (例如，真(TRUE)或假(FALSE))。取而代之，可使用 BOOL 数据类型的地址。

10.7 在代码段中编辑STL语句

10.7.1 用于语句表编程的设置

设置助记符

可在两套助记符之间进行选择：

- 德语
- 英语

在打开一个块之前，可在 SIMATIC 管理器中使用“语言”标签中的菜单命令**选项 > 自定义**来设置助记符。在编辑块时，无法改变助记符。

用户可在块属性对话框中编辑**块属性**。

在编辑器中，可打开一些块，并可根据需要对它们依次进行编辑。

10.7.2 STL语句的输入规则

在“用于 S7-300/400 – 编程块的语句表”手册或 STL 在线帮助中，可找到关于语句表编程语言表达式的描述(语言描述)。

当在增量输入模式下在 STL 中输入语句时，必须遵循下列基本原则：

- 对块进行编程时所采用的次序非常重要。在调用块之前，必须已经编写好所调用的块。
- 语句由标记(可选)、指令、地址、和注释(可选)组成。
  实例：　　M001：A　　I 1.0　　//注释
- 每条语句均单独占一行。
- 在一个块中，最多可输入 999 个程序段。
- 每个程序段最多可达到约 2000 行。如果进行放大或缩小，相应地，可输入更多或更少的行。
- 当输入指令或绝对地址时，无论是大写还是小写，将不进行任何区分。

10.8 更新块调用

10.8.1 更新块调用

可使用“LAD/STL/FBD － 编写S7块程序”中的菜单命令**编辑 > 块调用 > 更新**来自动更新已经无效的块调用。在发生下列接口变化之后，必须进行更新：

- 插入了新的形式参数
- 删除了形式参数
- 修改了形式参数的名称
- 改变了形式参数的类型
- 改变了形式参数的次序。

在分配形式参数和实际参数时，必须遵守以下指定次序的规则：

1. **相同的参数名称**：
   如果形式参数的名称仍然相同的话，实际参数将自动进行分配。
   特殊情况：在梯形图和功能块图中，如果数据类型(BOOL)相同，则用于二进制输入参数的先前链接将只能自动进行分配。如果数据类型已经变化，则先前的链接将作为一个打开的分支继续保留。
2. **相同的参数数据类型**：
   在分配名称相同的参数之后，尚未分配的实际参数将被分配给和“旧的”形式参数具有相同数据类型的形式参数。
3. **相同的参数位置**：
   在执行规则1和规则2之后，仍然没有分配的那些实际参数，按照它们在“旧的”接口中的参数位置，现在都将分配给形式参数。
4. 如果使用上述的三个规则仍然无法分配实际参数，那么，它们将被删除，或者，在梯形图或功能块图中的两个先前连接的情况下，它们将作为打开的分支继续保留。

在执行该功能之后，将检查在变量说明表以及程序代码段中所进行的修改。

10.8.2 改变接口

也可使用增量编辑器来修改已经使用 STEP 7、版本 5 进行编辑的离线块的接口：

1. 确保所有的块都已经使用STEP 7、版本5进行编译。为此，可生成一个用于所有块的源文件，并对该文件进行编译。
2. 修改相关块的接口。
3. 现在即可一个接一个地打开所有的调用块 －将以红色显示相应的调用。
4. 选择菜单命令**编辑 > 块调用 > 更新**。
5. 再次生成相关的背景数据块。

注意

如果在线打开的块的接口发生变化，则可能导致CPU 切换到STOP模式。重新整理块调用 首先修改已调用块的数量，然后执行重新整理功能以便与调用匹配。

10.9 保存逻辑块

10.9.1 保存逻辑块

为输入新创建的块或记录编程设备数据库中逻辑块代码段或说明表中的变化，必须保存各个块。数据随后将被写入到编程设备的硬盘中。

为将块保存在编程设备的硬盘中：

1. 激活希望保存块的工作窗口。
2. 选择以下菜单命令之一：
   - **文件 > 保存**将使用同一名称对块进行保存。
   - **文件 > 另存为**将使用一个不同的S7用户程序或一个不同的名称对快进行保存。在随后出现的对话框中输入新的路径或新的块名称。

在这两种情况中，只有在其语法没有任何错误时才可保存块。语法错误将在创建块时同时进行识别，然后以红色显示。在保存块之前，必须纠正这些错误。

注意

- 也可在SIMATIC管理器中将块或源文件保存在其它项目或库下面(例如，通过拖放操作)。
- 在SIMATIC管理器中，只能将块或完整的用户程序保存到存储卡。
- 如果在保存或编译大型块时出现问题，应重新组织项目。.为此，可使用SIMATIC管理器中的菜单命令**文件 > 重新组织**。然后尝试重新进行保存或编译。

---

11 创建数据块

11.1 创建数据块的基本信息

您可以在数据块(DB)中存储数值以便为机器或设备所访问。与采用梯形逻辑、语句表或功能块图表这些编程语言之一而编写的逻辑块相比，数据块只包含变量声明部分。这就表示此处与代码段不相关，因此只是编程段。

当您打开数据块时，既可以在声明视图中也可以在数据视图中查看块。您可以使用菜单命令**视图 > 声明视图**和**视图 > 数据视图**在这两种视图之间切换。

声明视图

在下列情况下可使用声明视图：

- 查看或确定共享数据块的数据结构，
- 查看带有相关的用户自定义数据类型(UDT)的数据块的数据结构，或者
- 查看带有相关功能块(FB)的数据块的数据结构。

不能修改与功能块或用户自定义的数据类型相关的数据块的结构。要修改它们，必须首先修改相关的FB或UDT，然后创建新的数据块。

数据视图

如果您想要修改数据，那么可以使用数据视图。您只能显示、输入或修改数据视图中每一个元素的实际值。在数据块的数据视图中，具有复杂数据类型的变量的元素分别以其全名列出。

背景数据块和共享数据块之间的差别

共享数据块不能分配给逻辑块。它包含设备或机器所需的值，并且可以在程序中的任何位置直接调用。

背景数据块是直接分配给逻辑块的数据块，如功能块。背景数据块包含存储在变量声明表的功能块中的数据。

11.2 数据块的声明视图

对于不是全局共享的数据块，不能改变声明视图。

列	解释
地址	显示在您完成声明的输入后由STEP 7自动为变量分配的地址。
声明	仅背景数据块会显示该列。它显示功能块变量声明中的变量是如何声明的： • 输入参数(IN) • 输出参数(OUT) • 输入/输出参数(IN_OUT) • 静态数据(STAT)
名称	此处输入必须分配给每个变量的符号名。
类型	输入想要分配给变量的数据类型(BOOL、INT、WORD、ARRAY等)。变量可以具有基本数据类型、复杂数据类型或者用户自定义的数据类型。
初始值	如果您不想让软件使用缺省的数据类型，可在此处输入初始值。所有的值都必须与数据类型兼容。 当您首次保存块时，如果您没有明确定义变量的实际值，那么，初始值将作为当前值使用。 请注意：初始值不能下载给CPU。
注释	在该域中输入一条注释有助于对变量进行归档。注释可以多达80个字符。

11.3 数据块的数据视图

数据视图显示数据块中所有变量的当前值。您只能在数据视图中改变这些值。对于所有共享数据块，该视图中的表格表示均相同。对于背景数据块，将显示一个附加的“声明”列。

对于有复杂数据类型或用户自定义数据类型的变量，所有元素均将以其在数据视图中的完整符号名称显示在它们自己的行中。如果这些元素均位于背景数据块的IN_OUT 区，那么，指针将指向“实际值”列中的复杂或用户自定义数据类型。

数据视图将显示下列列：

列	解释
地址	显示STEP 7为变量自动分配的地址。
声明	仅背景数据块会显示该列。它显示功能块变量声明中的变量是如何声明的： • 输入参数(IN) • 输出参数(OUT) • 输入/输出参数(IN_OUT) • 静态数据(STAT)
名称	在变量声明中为变量分配的符号名称。在数据视图中不能编辑该域。

列	解释
类型	显示为变量定义的数据类型。 对于共享数据块，这里将只列出基本数据类型，因为对于具有复杂或用户自定义数据类型的变量，其元素将在数据视图中逐个列出。 对于背景数据块，还会显示参数类型，对于具有复杂或用户自定义数据类型的输入/输出参数(IN_OUT)，指针将指向“实际值”列中的数据类型。
初始值	如果您不希望软件使用指定数据类型的缺省值时，可为变量输入初始值。 当您首次保存数据块时，如果没有明确定义变量的实际值，那么，将使用初始值作为当前值。 请注意：与实际值不同，初始值不能下载给CPU。
实际值	离线：打开数据块时的变量值，或上一次修改、保存过的变量值(即使在线打开数据块，该显示也不进行更新)。 在线：打开数据块时的当前值，显示但不自动更新。如要更新视图，请按F5。 如果它不属于具有复杂或用户自定义数据类型的输入/输出参数(IN_OUT)，您可编辑该域。所有的值都必须与数据类型兼容。 请注意：仅当前的值才可下载给CPU/
注释	所输入的用来对变量归档的注释。在数据视图中不能编辑该域。

11.4 编辑和保存数据块

11.4.1 输入共享数据块的数据结构

如果打开一个未分配给用户自定义数据类型或功能块的数据块，则可以在数据块的声明视图中定义其结构。对于不共享的数据块，不能改变声明视图。

1. 打开一个共享数据块，表示该块与UDT或FB无关。
2. 如果该视图尚未设置，则显示数据块的声明视图。
3. 根据下列信息填写所显示的表格以定义结构。

对于未共享的数据块，不能修改声明视图。

列	解释
地址	显示在您完成声明的输入后由STEP 7自动为变量分配的地址。
名称	此处输入必须分配给每个变量的符号名。
类型	输入想要分配给变量的数据类型(BOOL、INT、WORD、ARRAY等)。变量可以具有基本数据类型、复杂数据类型或者用户自定义的数据类型。
初始值	如果您不想让软件使用缺省的数据类型，可在此处输入初始值。所有的值都必须与数据类型兼容。 当您第一次保存块时，如果还没有为变量明确定义实际值，那么该初始值将用作实际值。
注释	在该域中输入可选的注释以对变量进行归档。注释可以多达80个字符。

11.4.2 输入和显示引用FB (背景数据块)的数据块的数据结构

输入

在将数据块与功能块(背景数据块)相关联时，该功能块的变量声明定义了数据块的结构。任何改动都只能在相关的功能块中进行。

1. 打开相关的功能块(FB)。
2. 编辑功能块的变量声明。
3. 再次创建背景数据块。

显示

在背景数据块的声明视图中，您可以显示如何对功能块中的变量进行声明。

1. 打开数据块。
2. 如果该视图尚未设置，则显示数据块的声明视图。
3. 参见下表以获取更多信息。

对于不共享的数据块，不能改变声明视图。

列	解释
地址	显示STEP 7为变量自动分配的地址。
声明	该列显示了在功能块的变量声明中是如何对变量进行声明的： • 输入参数(IN) • 输出参数(OUT) • 输入/输出参数(IN_OUT) • 静态数据(STAT) 功能块的已声明的临时本地数据不在背景数据块中。
名称	在功能块的变量声明中分配的符号名。
类型	显示功能块的变量声明中分配的数据类型。变量可以具有基本数据类型、复杂数据类型或者用户自定义的数据类型。 如果在已经声明了调用静态变量的功能块中调用了附加功能块，那么也可以在此处将功能块或系统函数块(SFB)指定为数据类型。
初始值	如果不想让软件使用缺省值，可以在功能块的变量声明中为变量输入初始值。 当您第一次保存数据块时，如果尚未为该变量明确定义实际值，那么该初始值将用作实际值。
注释	在该功能块的变量声明中所输入的注释用于对数据元素进行归档。您不能编辑该域。

注意

对于分配给功能块的数据块，您只能编辑变量的实际值。要输入变量的实际值，必须在数据块的数据视图中进行操作。

11.4.3 输入用户自定义数据类型(UDT)的数据结构

1. 打开用户自定义的数据类型(UDT)。
2. 如果该视图尚未设置，则显示声明视图。
3. 通过确定变量的顺序、数据类型和初始值来定义UDT的结构，并可以在需要时使用下表中的信息。
4. 通过使用TAB键或RETURN建退出该行，以完成变量的输入。

列	解释
地址	显示在您完成声明的输入后由STEP 7自动为变量分配的地址。
名称	此处输入必须分配给每个变量的符号名。
类型	输入想要分配给变量的数据类型(BOOL、INT、WORD、ARRAY等)。变量可以具有基本数据类型、复杂数据类型或者用户自定义的数据类型。
初始值	如果您不想让软件使用缺省的数据类型，可在此处输入初始值。所有的值都必须与数据类型兼容。 当您第一次保存用户自定义的数据类型(或变量、数据块)的实例时，如果尚未为变量明确定义实际值，那么该初始值将用作实际值。
注释	在该域中输入注释将有助于对变量进行归档。注释可以多达80个字符。

11.4.4 输入和显示引用UDT的数据块的结构

输入

当您将数据块分配给用户自定义的数据类型时，该用户自定义数据类型的数据结构即定义了数据块的结构。任何改动都只能在相应的用户自定义数据类型中进行。

1. 打开用户自定义的数据类型(UDT)。
2. 编辑用户自定义数据类型的结构。
3. 再次创建数据块。

显示

您只能在数据块的声明视图中显示如何在用户自定义的数据类型中对变量进行声明。

1. 打开数据块。
2. 如果该视图尚未设置，则显示数据块的声明视图。
3. 参见下表以获取更多信息。

不能修改声明视图。任何改动都只能在相应的用户自定义数据类型中进行。

列	解释
地址	显示STEP 7为变量自动分配的地址。
名称	在用户数据类型的变量声明中分配的符号名。
类型	显示用户自定义数据类型的变量声明中分配的数据类型。变量可以具有基本数据类型、复杂数据类型或者用户自定义的数据类型。
初始值	如果不想让软件使用缺省值，可以为该变量输入用户自定义数据类型的初始值。 当您第一次保存数据块时，如果尚未为该变量明确定义实际值，那么该初始值将用作实际值。
注释	在变量声明中为用户自定义数据类型所输入的注释用于对数据元素进行归档。

注意

对于分配给用户自定义数据类型的数据块，您只能编辑该变量的实际值。要输入变量的实际值，必须在数据块的数据视图中进行操作。

11.4.5 在数据视图中编辑数据值

只能在数据块的数据视图中编辑实际值。

1. 如果需要，使用菜单命令**视图 > 数据视图**切换到数据视图中的表格显示。
2. 在“实际值”列的域中输入数据元素所需的实际值。实际值必须与数据元素的数据类型相兼容。

在编辑期间，任何错误的输入(例如，如果输入的实际值与数据类型不兼容)都会立即被识别并以红色显示。这些错误必须在保存数据块之前予以更正。

注意

对数据值的任何改变仅在保存了数据块时才被保留。

11.4.6 重新设置数据值为其初始值

只能在数据块的数据视图中重新设置数据值。

1. 如果需要，使用菜单命令**视图 > 数据视图**切换到数据视图中的表格显示。
2. 为此，选择菜单命令**编辑 > 初始化数据块**。

所有变量将被重新分配其期望的初始值，这表示所有变量的实际值将由其对应的初始值所覆盖。

注意

对数据值的任何改变仅在保存了数据块时才被保留。

11.4.7 保存数据块

为将新创建的块或数据块中经修改的数据值输入到编程设备数据库中，必须保存各个块。数据随后将被写入到编程设备的硬盘中。

为将块保存在编程设备的硬盘中：

1. 激活希望保存块的工作窗口。
2. 选择以下菜单命令之一：
   - **文件 > 保存**将使用同一名称对块进行保存。
   - **文件 > 另存为**将使用一个不同的S7用户程序或一个不同的名称对快进行保存。在随后出现的对话框中输入新的路径或新的块名称。对于数据块，由于名称DB0已为系统保留，所以可能无法使用该编号。

在这两种情况中，只有在其语法没有任何错误时才可保存块。语法错误将在创建块时同时进行识别，然后以红色显示。在保存块之前，必须纠正这些错误。

注意

- 也可在SIMATIC管理器中将块或源文件保存在其它项目或库下面(例如，通过拖放操作)。
- 在SIMATIC管理器中，只能将块或完整的用户程序保存到存储卡。
- 如果在保存或编译大型块时出现问题，应重新组织项目。为此，可使用SIMATIC管理器中的菜单命令**文件 > 重新组织**。然后尝试重新进行保存或编译。

---

12 为数据块分配参数

12.1 将参数分配给数据块

“数据块的参数分配”功能使您能够在 LAD/STL/FBD 程序编辑器以外完成下列操作：

- 编辑和下载背景数据块的实际值给 PLC，无需装载整个数据块。
- 在线监视背景数据块。
- 使用“S7_techparam”系统属性(技术功能)来轻松地将参数分配给背景数据块和多重背景，并对其进行在线监视。

过程：

1. 在SIMATIC管理器中，双击、打开背景数据块。
2. 如果希望打开“数据块的参数分配”功能，出现提问时请回答“是”。**结果**：背景数据块在“数据块的参数分配”应用中打开。
3. 通过选择菜单命令**视图 > 数据视图**或**视图 > 声明视图**，选择将在其中显示数据块的视图。对于具有“S7_techparam”系统属性的背景数据块或多重背景而言，“技术参数”视图将自动打开。
4. 根据需要编辑背景数据块。任何相关的信息、警告或错误都将显示在消息窗口中。为了跳转到出现警告或错误的位置，可双击相应的警告或错误。
5. 将已经修改的实际值从编程设备(PG)下载到您已经分配给当前S7程序的CPU (菜单命令**PLC > 下载参数设置数据**)。
6. 选择菜单命令**调试 > 监视**，显示已打开块的程序状态，然后在线监视所装载的实际值的编辑。

注意

您可识别具有“S7_techparam”系统的数据块。为确定一个块是否具有该系统属性，可跳转到SIMATIC管理器，并选择块。然后，选择菜单命令**编辑 > 对象属性**，并打开“属性”标签。

12.2 分配参数给技术功能

使用“为数据块分配参数”功能，可以很容易地将参数分配给标准库所提供的温度控制器块 FB 58“TCONT_CP”和 FB 59“TCONT_S”，并可以在线监控它们。

为此，可如下进行操作：

1. 在SIMATIC管理器中，选择菜单命令**文件 > 打开 > 库**，打开STEP 7标准库。
2. 选择“PID控制块”，然后点击“块”。此处，将发现具有属性“S7_techparam”的下列功能块：
   - FB 58“TCONT_CP”：执行器温度控制器，具有连续或脉冲输入信号
   - FB 59“TCONT_S”：整型执行器的温度控制器
3. 将合适的功能块(FB 58或FB 59)从标准库复制到项目。
4. 选择菜单命令**插入 > S7块 > 数据块**，为所选择的FB创建背景数据块。
5. 在SIMATIC管理器中，双击背景数据块，打开，并启动“数据块的参数分配”功能。
   结果：背景数据块在技术视图中打开。现在可以很容易地分配参数给背景数据块，并在线监视它。
6. 在技术视图中输入合适的控制器值。任何相关的信息、警告或错误都将显示在消息窗口中。为了跳转到出现警告或错误的位置，可双击相应的警告或错误。

注意

在SIMATIC管理器中，选择块，再选择菜单命令**编辑 > 对象属性**，然后打开“属性”标签，可以确定块是否具有“S7_techparam”系统属性。

---

13 创建STL源文件

13.1 关于在STL源文件中编程的基本信息

您可输入程序或其中的一部分作为 STL 源文件，然后在一步之内将其编译成块。源文件可包含许多块的代码，它们随后将在一次编译运行中编译为块。

使用源文件创建程序将具有下列优点：

- 可使用任意的 ASCII 编辑器创建和编辑源文件，然后使用该应用程序将其导入并编译成块。编译过程将创建单个的块并将其存储在 S7 用户程序中。
- 您可在一个源文件中对许多块进行编程。
- 即使包含有语法错误，也可保存源文件。而如果您使用增量语法检查创建逻辑块，就无法做到。然而，一旦您对源文件进行编译，只报告语法错误。

源文件将按照编程语言表达式语句表(STL)的语法进行创建。源文件将给出其块结构、变量声明、以及使用关键字的网络。

当创建 STL 源文件中的块时，您应注意如下几点：

- STL 源文件的编程准则
- STL 源文件中块的语法和格式
- STL 源文件中块的结构

13.2 STL源文件中的编程规则

13.2.1 在STL源文件中输入语句的规则

STL 源文件主要包含连续的文本。为了让文件能够编译成块，必须遵守特定的结构和语法规则。

在将用户程序创建为 STL 源文件时应用下列通则：

议题	规则
语法	STL语句的句法与增量语句表编辑器相同。其中有一个例外就是CALL指令。
CALL	在源文件中，在括号中输入参数。各个参数之间用逗号分隔。 实例：FC调用(一行) CALL FC10(param1 :=I0.0,param2 :=I0.1); 实例：FB调用(一行) CALL FB10, DB100(para1 :=I0.0,para2 :=I0.1); 实例：FB调用(多行) CALL FB10, DB100(para1 :=I0.0, para2 :=I0.1); 注意： 在调用块时，按照参数在ASCII编辑器中定义的次序进行传送。否则，这些行的注释分配将在STL和源文件视图中不匹配。
大/小写	该应用程序中的编辑器不区分大小写，但是系统属性和跳转标签例外。在输入字符串时(数据类型为STRING)，您也必须注意大小写。 关键字以大写显示。在编译时，无需遵守大小写；因此您可以按大写或小写输入关键字，也可以将大小写混合使用。
分号	用分号(;)指示每一STL语句和每个变量声明的结束。每行可以输入多个语句。
双斜杆(//)	每个注释都以双斜杆(//)开始，以回车(或换行)符结束。

13.2.2 在STL源文件中声明变量的规则

对于源文件中的每个块，必须声明所需的变量。

变量声明部分位于块编码部分的前面。

变量(如果正被使用)必须按声明类型的正确顺序进行声明。这表示同一种声明类型的所有变量都集中在一起。

对于梯形图、功能块图和语句表，只要填写一张变量声明表，但是在此必须使用相关的关键字。

用于变量声明的关键字

声明类型	关键字	适用于...
输入参数	"VAR_INPUT" 声明列表 "END_VAR"	FB、FC
输出参数	"VAR_OUTPUT" 声明列表 "END_VAR"	FB、FC
输入/输出参数	"VAR_IN_OUT" 声明列表 "END_VAR"	FB、FC
静态变量	"VAR" 声明列表 "END_VAR"	FB
临时变量	"VAR_TEMP" 声明列表 END_VAR	OB、FB、FC

关键字 END_VAR 指示声明列表的结束。

声明列表是一个声明类型的变量列表，可以在其中将缺省值分配给变量(例外：VAR_TEMP)。下列实例显示了声明列表中某个条目的结构：

Duration_Motor1 ：	**S5TIME**	**：=**	**S5T#1H_30M**	**；**
变量	**数据类型**		**缺省值**	

注意

- 变量符必须以字母开头。不能给变量分配一个与所保留的关键字相同的符号名。
- 如果本地声明和符号表中有相同的变量符，可以通过在本地变量名称前面放置#，并在符号表中将变量加上引号，从而对本地变量进行编码。否则，块将这些变量解释为本地变量。

13.2.3 在STL源文件中块次序的规则

被调用的块位于对其调用的块之前。这表示：

- 在大多数情况下用到的 OB1，由于它要调用其它所有块，因而位于最后。被 OB1 调用的块必须位于其前面。
- 用户自定义的数据类型(UDT)位于用到它们的那些块之前。
- 带有相关的用户自定义的数据类型(UDT)的数据块跟在用户自定义的数据类型之后。
- 共享数据块位于所有调用它们的块之前。
- 背景数据块跟在相关的功能块之后。
- DB0 是保留块。不能创建具有该名称的数据块。

13.2.4 在STL源文件中设置系统属性的规则

可以将系统属性分配给块和参数。它们控制消息组态和链接组态、操作员界面功能以及过程控制组态。

当在源文件中输入系统属性时，应用以下规则：

- 用于系统属性的关键字总是以 S7_开始。
- 系统属性放在括号中(大括号)。
- 语法：{S7_identifier := 'string'}
  许多标识符用“；”分隔。
- 用于块的系统属性位于块属性之前，关键字 ORGANIZATION_和 TITLE 之后。
- 用于参数的系统属性包括在参数声明中，表示它们位于数据声明的冒号之前。
- 大小写字符之间有区别。这表示在输入系统属性时，正确使用大小写字符是很重要的。

用于块的系统属性可以在增量输入模式中使用“属性”标签页下的菜单命令**文件 > 属性**进行检查和修改。

用于参数的系统属性可以在增量输入模式中使用菜单命令**编辑 > 对象属性**进行检查和修改。光标必须置于参数声明的名称域中。

13.2.5 在STL源文件中设置块属性的规则

如果使用块属性，可以更容易地识别您创建的块，还可以保护这些块免受未授权地更改。

可以使用菜单命令**文件 > 属性**在“常规 - 第 1 部分”和“常规 - 第 2 部分”标签页中以增量输入模式检查或更改块属性。

其它块属性只能输入到源文件中。

以下规则适用于源文件：

- 块属性位于变量声明部分之前。
- 每个块属性都具有属于自己的行。
- 每行以分号结束。
- 块属性使用关键字进行指定。
- 如果输入块属性，它们必须按块属性表中的顺序显示。
- 适用于每种块类型的块属性在“分配：块属性给块类型”中列出。

注意

块属性还显示在SIMATIC管理器用于块的对象属性中。也可以在此编辑属性AUTHOR、FAMILY、NAME和VERSION。

块属性和块次序

在输入块属性时，应遵循下表中说明的输入顺序：

次序	关键字/属性	含义	实例
1.	[KNOW_HOW_PROTECT]	块保护；使用此选项编译的块将不能查看其代码段。可以查看块的接口，但不能更改。	KNOW_HOW_PROTECT
2.	[AUTHOR:]	作者名：公司名、部门名或其它名称 (最多8个不含空格的字符)	AUTHOR：西门子公司，但无关键字
3.	[FAMILY:]	块系列的名称：例如，控制器 (最多8个不含空格的字符)	FAMILY：控制器，但无关键字
4.	[NAME:]	块名称(最多8个字符)	NAME：PID，但无关键字
5.	[VERSION:int1 . int2]	块的版本号 (两个数都介于0和15之间，即0.0至15.15)	VERSION : 3.10
6.	[CODE_VERSION1]	指示功能块是否可声明多重背景的标识号。如果想声明多重背景，则功能块不可具有此属性	CODE_VERSION1
7.	[UNLINKED]仅适用于DB	具有UNLINKED属性的数据块只存储在装入存储器中。它们不占用任何工作存储器空间，并且不与程序链接。不能使用MC7命令访问它们。此类DB的内容只能使用SFC 20 BLKMOV (S7-300. S7-400)或SFC 83 READ_DBL (S7-300C)传送给工作存储器。	
8.	[READ_ONLY]仅适用于DB	用于数据块的写保护；其数据只能读取，不能修改	FAMILY= Examples VERSION= 3.10 READ_ONLY

13.2.6 每个块类型允许的块属性

下表说明块类型及其对应的可声明块属性：

属性	OB	FB	FC	DB	UDT
KNOW_HOW_PROTECT	•	•	•	•	–
AUTHOR	•	•	•	•	–
FAMILY	•	•	•	•	–
NAME	•	•	•	•	–
VERSION	•	•	•	•	–
UNLINKED	–	–	–	•	–
READ_ONLY	–	–	–	•	–

用 KNOW_HOW_PROTECT 设置块保护

当您在 STL 源文件中编写块时，可以通过使用关键字 KNOW_HOW_PROTECT 设置块保护，以防止您的块受到未授权用户的访问。

块保护将导致以下结果：

- 如果想在稍后阶段在 STL、FBD 或梯形图增量编辑器中查看已编译的块，将无法显示块的代码段。
- 用于块的变量声明列表只显示声明类型 var_in、var_out 和 var_in_out 的变量。声明类型为 var_stat 和 var_temp 的变量保持隐藏状态。
- 关键字 KNOW_HOW_PROTECT 在其它块属性之前输入。

用 READ_ONLY 为数据块设置写保护

可以为数据块设置写保护，以便在编程期间不会将这些块覆盖。为此，数据块必须以 STL 源文件的形式存在。

在源文件中使用关键字 READ_ONLY 设置写保护。该关键字必须立即出现在变量声明的该行之前。

13.3 STL源文件中块的结构

使用关键字对STL源文件中的块进行结构化。根据块的类型，在下列结构上有所区别：

- 逻辑块
- 数据块
- 用户自定义的数据类型(UDT)

13.3.1 STL源文件中逻辑块的结构

逻辑块由以下部分组成，每个部分都用相应的关键字进行识别：

- 块起始，
- 通过关键字和块编号或者块名称进行识别，例如
  - “ORGANIZATION_BLOCK OB1”用于组织块，
  - “FUNCTION_BLOCK FB6”用于功能块，或者
  - “FUNCTION FC1 :INT”用于函数。对于函数，还要指定函数类型。这可以是基本数据类型或者复杂数据类型(除了 ARRAY 和 STRUCT)，并定义返回值(RET_VAL)的数据类型。如果没有返回任何值，就给出关键字 VOID。
- 可选的块标题用关键字“TITLE”引入(标题的最大长度：64 个字符)
- 附加的注释在行的起始处用双斜杆开始
- 块属性(可选)
- 变量声明部分
- 代码段，以“BEGIN”开始。代码段包括一个或多个程序段，它们用“NETWORK”标识。不能输入程序段编号。
- 对于每个所使用程序段的可选程序段，用关键字“TITLE =”引入(标题的最大长度：64 个字符)
- 每个程序段的附加注释在行的起始处用双斜杆开始
- 块结束，用 END_ORGANIZATION_BLOCK、END_FUNCTION_BLOCK 或 END_FUNCTION 标识
- 在块类型和块编号之间必须放置一个空格。符号块名称可以通过引号来标识，以确保本地变量的符号名和符号表中的名称保持唯一性。

13.3.2 STL源文件中数据块的结构

数据块包括以下区域，它们分别由其相应的关键字引入：

- 块起始，用关键字和块编号或块名称进行标识，例如 DATA_BLOCK DB26
- 引用相关的 UDT 或功能块(可选)
- 由关键字 TITLE =引入的可选块标题(超出 64 个字符的部分将被剪切)
- 可选的块注释，以双斜杆//开始
- 块属性(可选)
- 变量声明部分(可选)
- 分配缺省值的部分，以 BEGIN (可选)开始
- 块结束，通过 END_DATA_BLOCK 进行标识

有三种类型的数据块：

- 数据块，用户自定义
- 具有相关的用户自定义数据类型(UDT)的数据块
- 具有相关功能块的数据块(例如“背景”数据块)

13.3.3 STL源文件中用户自定义数据类型的结构

用户自定义的数据类型包括以下区域，它们分别由其相应的关键字引入：

- 块起始，用关键字 TYPE 和编号或名称进行标识，例如 TYPE UDT20
- 结构化的数据类型
- 块结束，用 END_TYPE 标识

当您输入用户自定义的数据类型时，必须确保用户自定义的数据类型位于使用它们的块之前。

13.4 STL源文件中块的语法和格式

格式表显示了在编写 STL 源文件时应该遵守的语法和格式。语法表示如下：

- 每个元素在右列中描述。
- 必须输入的所有元素都用引号显示。
- 方括号[...]表示这些括号中的内容是可选的。
- 关键字用大写字母给出。

13.4.1 组织块的格式表

下表简短地列出了在 STL 源文件中用于组织块的格式：

结构	描述
“ORGANIZATION_BLOCK” ob_no或ob_name	ob_no是块编号，例如：OB1； ob_name是在符号表中定义的块的符号名
[TITLE=]	块标题(超出64个字符长度的部分将被剪切)
[块注释]	可以在“//”之后输入注释
[用于块的系统属性]	用于块的系统属性
[块属性]	块属性
变量声明部分	临时变量的声明
“BEGIN”	该关键字用于将变量声明部分从STL指令的列表中分隔开
NETWORK	程序段的开始
[TITLE=]	程序段标题(最多64个字符)
[程序段注释]	可以在“//”之后输入注释
STL指令的列表	块指令
“END_ORGANIZATION_ BLOCK”	该关键字用于结束组织块

13.4.2 功能块的格式表

下表简短地列出了在 STL 源文件中用于功能块的格式：

结构	描述
“FUNCTION_BLOCK”fb_no或fb_name	fb_no是块编号，例如FB6； fb_name是在符号表中定义的块的符号名
[TITLE=]	块标题(超出64个字符长度的部分将被剪切)
[块注释]	可以在“//”之后输入注释
[用于块的系统属性]	用于块的系统属性
[块属性]	块属性
变量声明部分	输入、输出和输入/输出参数、以及临时或静态变量的声明 参数声明也可能包含用于参数的系统属性的声明。
“BEGIN”	该关键字用于将变量声明部分从STL指令的列表中分隔开
NETWORK	程序段的开始
[TITLE=]	程序段标题(最多64个字符)
[程序段注释]	可以在“//”之后输入注释
STL指令的列表	块指令
“END_FUNCTION_BLOCK”	该关键字用于结束功能块

13.4.3 功能的格式表

下表简短地列出了在 STL 源文件中用于功能的格式：

结构	描述
“FUNCTION” fc_no:fc_type或 fc_name:fc_type	fb_no是块编号，例如FC5； fb_name是在符号表中定义的块的符号名 fc_type是功能返回值(RET_VAL)的数据类型。这可以是基本数据类型或复杂数据类型(除了ARRAY和STRUCT)或VOID。 如果想要使用系统属性 作为返回值(RET_VAL)，必须为数据声明冒号前的参数输入系统属性。
[TITLE=]	块标题(超出64个字符长度的部分将被剪切)
[块注释]	可以在“//”之后输入注释
[用于块的系统属性]	用于块的系统属性
[块属性]	块属性

结构	描述
变量声明部分	输入、输出和输入/输出参数、以及临时变量的声明
“BEGIN”	该关键字用于将变量声明部分从STL指令的列表中分隔开
NETWORK	程序段的开始
[TITLE=]	程序段标题(最多64个字符)
[程序段注释]	可以在“//”之后输入注释
STL指令的列表	块指令
“END_FUNCTION”	该关键字用于结束功能块

13.4.4 数据块的格式表

下表简短地列出了在 STL 源文件中用于数据块的格式：

结构	描述
“DATA_BLOCK”db_no或 db_name	db_no是块编号，例如DB5； db_name是在符号表中定义的块的符号名
[TITLE=]	块标题(超出64个字符长度的部分将被剪切)
[块注释]	可以在“//”之后输入注释
[用于块的系统属性]	用于块的系统属性
[块属性]	块属性
声明部分	声明该块是与UDT还是FB有关，给出符号表中定义的块编号或符号名，或者复杂数据类型。
“BEGIN”	该关键字用于将变量声明部分从数值分配列表中分隔开
[分配初始值]	可以为变量分配指定的初始值。各变量或是具有分配的常数，或是具有至其它块的引用。
“END_DATA_BLOCK”	该关键字用于结束数据块

13.5 创建STL源文件

13.5.1 创建STL源文件

必须在 S7 程序的源文件文件夹中创建源文件。可以在 SIMATIC 管理器或编辑器窗口中创建源文件。

在SIMATIC管理器中创建源文件

1. 通过双击相应的“源文件”文件夹将其打开。
2. 要插入STL源文件，可选择菜单命令**插入 > S7软件 > STL源文件**。

在编辑器窗口中创建源文件

1. 选择菜单命令**文件 > 新建**。
2. 在对话框中选择相同S7程序的源文件文件夹，其中包含带有这些块的用户程序。
3. 输入新的源文件的名称。
4. 使用“确定”进行确认。

源文件以您所输入的名称创建，并显示在用于编辑的窗口中。

13.5.2 编辑S7源文件

程序语言和用于编辑源文件的编辑器可以在源文件的对象属性中设置。这可确保在打开源文件进行编辑时，就启动了正确的编辑器和正确的编程语言。STEP 7 标准软件包支持在STL源文件中编程。

其它程序语言也可用作可选软件包。如果在您的计算机上已经加载了相应的软件选件，那么只需选择菜单命令便可以插入源文件。

要编辑 S7 源文件，可如下操作：

1. 通过双击相应的“源文件”文件夹将其打开。
2. 如下启动进行编辑时所需的编辑器：
   - 在窗口的右半部分双击所需的源文件。
   - 在窗口的右半部分选择所需的源文件，然后选择菜单命令**编辑 > 打开对象**。

13.5.3 设置源代码文本的布局

要提高源文件中文本的可读性，请选择菜单命令**选项 > 设置**，然后选择“源代码”标签。为源代码的各种元素指定字体、字体风格以及颜色。

例如，可以指定按大写字母显示行号和关键字。

13.5.4 在STL源文件中插入块模板

用于组织块(OB)、功能块(FB)、功能(FC)、数据块(DB)、背景数据块、带有相关的用户自定义数据类型的数据块以及用户自定义数据类型(UDT)的块模板可用于 STL 源文件中的编程。使用块模板，可以轻而易举地将块插入到您的源文件中，同时遵守语法和结构准则。

要插入块模板，可如下操作：

1. 激活想要在其中插入块模板的源文件的窗口。
2. 将光标放置在文件中想要插入块模板的位置处。
3. 选择菜单命令插入 > 块模板 > OB/FB/FC/DB/背景数据块/DB参考UDT/UDT之一。

块模板插入到文件中光标位置的后面。

13.5.5 插入其它STL源文件的内容

可以将其它源文件的内容插入到您的 STL 源文件中。

操作过程如下：

1. 激活想要插入其它源文件内容的源文件窗口。
2. 将光标放置在文件中想要在其后插入源文件的位置处。
3. 选择菜单命令**插入 > 对象 > 文件**。
4. 在出现的对话框中选择所需的源文件。

选择的源文件的内容插入到光标位置之后。保留换行(回车)。

13.5.6 在STL源文件中插入来自现有块的源代码

可以将来自其它块的源代码插入您的 STL 源文件中，该文件是以梯形图、功能块图或语句表创建的。这也可以用于组织块(OB)、功能块(FB)、功能(FC)、数据块(DB)以及用户自定义的数据类型(UDT)。

操作过程如下：

1. 激活想要在其中插入块的源文件的窗口。
2. 将光标放置在文件中想要在其后插入来自块的源代码的位置处。
3. 选择菜单命令**插入 > 对象 > 块**。
4. 在出现的对话框中选择所需的块。

从该块中生成等效的源文件。将源文件的内容插入到该光标位置后。

13.5.7 插入外部源文件

可以用 ASCII 编辑器创建和编辑源文件，然后使用该应用程序将其导入项目并编译为各个块。为此，您必须将源文件导入 S7 程序的“源文件”文件夹中，在该 S7 程序中将保存编译期间所创建的块。

要插入外部源文件，可如下操作：

1. 选择要导入外部源文件的S7程序的源文件的文件夹。
2. 选择菜单命令插入 > 外部源文件。
3. 在出现的对话框中，输入您想要导入的源文件。

您正在导入的源文件的文件名必须具有有效的文件扩展名。STEP 7使用文件扩展名来确定源文件类型。这就是说，例如STEP 7在导入带有扩展名.AWL的文件时将创建STL源文件。有效的文件扩展名在对话框的“文件类型”下列出。

注意

还可以使用菜单命令**插入 > 外部源文件**导入使用STEP 7版本1创建的源文件。

13.5.8 生成来自块的STL源文件

您可以生成 STL 源文件，该文件可以用来自现有块的任一文本编辑器进行编辑。在 S7 程序的源文件的文件夹中生成源文件。

要从块中生成源文件，可如下操作：

1. 在程序编辑器中，选择菜单命令**文件 > 生成源文件**。
2. 在该对话框中，选择想要在其中创建新的源文件的文件夹。
3. 在文本框中输入源文件的名称。
4. 在“选择STEP 7块”对话框中，选择想要生成为指定源文件的块。所选块显示在右列表框中。
5. 点击“确定”，进行确认。

从所选块中创建一个连续的 STL 源文件，并将其显示在窗口中以进行编辑。

13.5.9 导入源文件

要将任一目录下的源文件导入到项目中：

1. 在SIMATIC管理器中，选择要导入源文件的源文件文件夹。
2. 选择菜单命令插入 > 外部源文件。
3. 在显示的对话框中，选择目标目录和要导入的源文件。
4. 点击“打开”按钮。

13.5.10 导出源文件

从一个项目将源文件导出到任一个目标目录：

1. 在源文件夹中选择源文件。
2. 在SIMATIC管理器中选择菜单命令编辑 > 导出源文件。
3. 在所显示的对话框中输入目标目录和文件名。
4. 点击“保存”按钮。

注意

如果对象名称没有文件扩展名，则从文件类型获取的文件扩展名会添加到文件名称上。例如，STL源文件“prog”导出到文件“prog.awl”中。

如果对象名称已经有一个有效的文件扩展名，那么保持该扩展名不变。例如，STL源文件“prog.awl”导出到文件“prog.awl”中。

如果对象名称具有无效的文件扩展名(在名称中包含英文句号)，那么不添加文件扩展名。

可在“导出源文件”对话框的“文件类型”下获得有效文件扩展名的列表。

13.6 保存与编译STL源文件并执行一致性检查

13.6.1 保存STL源文件

可以在任何时候保存当前状态下的 STL 源文件。程序未编译并且没有执行语法检查，这表示同时将错误也进行了保存。

仅在编译源文件时或者进行一致性检查之后，检测并报告语法错误。

要以相同的名称保存源文件：

1. 激活想要保存的源文件的窗口。
2. 选择菜单命令**文件 > 保存**。

要以新名称保存源文件或者将源文件保存到其它项目中：

1. 激活想要保存的源文件的窗口。
2. 选择菜单命令**文件 > 另存为**。
3. 在对话框中，选择想要用来保存该源文件的文件夹并输入文件名。

13.6.2 检查STL源文件中的一致性

使用菜单命令**文件 > 一致性检查**可以显示STL源文件中的所有语法错误。与编译相比，将不生成任何块。

在完成一致性检查后，将出现一个对话框向您显示找到的总错误数。

在窗口下半部分别列出找到的所有错误，并带有行参考。在编译源文件之前纠正这些错误，以便可以创建所有的块。

13.6.3 调试STL源文件

激活的源文件窗口被分为两个部分。下列错误在下半部分中列出：

- 编译后找到的错误通过菜单命令**文件 > 编译**启动。
- 一致性检查后找到的错误通过菜单命令**文件 > 一致性检查**启动。

要在源文件中找到错误位置，可将光标定位在消息窗口的“错误”标签上。错误的元素将在代码段自动高亮显示，并且在状态栏处将输出错误消息。

13.6.4 编译STL源文件

要求

为了能够将您在源文件中创建的程序编译成块，必须满足以下要求：

- 只能对保存在 S7 程序下的“源文件”文件夹中的源文件进行编译。
- 与“源文件”文件夹一样，“块”文件夹也必须位于 S7 程序下，在该 S7 程序中可以保存编译期间所创建的块。只有在正确无误地编译了源文件后，才能创建在源文件中编写的块。如果在源文件中编写了若干块，则只会创建没有错误的块。于是您可以打开这些块、对其进行编辑、将它们下载到 CPU、以及逐个调试。

编辑器中的过程

1. 打开想要编译的源文件。源文件必须位于S7程序的源文件的文件夹中，在该S7用户程序中将保存所编译的块。
2. 选择菜单命令**文件 > 编译**。
3. 显示“编译器报告”对话框，在其中将显示所编译的行数和找到的语法错误。

仅当正确无误地编译了源文件后，才能为文件创建指定的块。如果在源文件中编写了若干块，则只会创建没有错误的块。错误警告不会阻止块的创建。

编译期间检测到的语法错误将显示在工作窗口的下半部分，并且必须在更正后才能创建相应的块。

SIMATIC管理器中的过程

1. 通过双击相应的“源文件”文件夹将其打开。
2. 选择想要编译的一个或多个源文件。不能为关闭的源文件文件夹启动编译，以编译它下面包含的所有源文件。
3. 选择菜单命令**文件 > 编译**以启动编译。为所选的源文件调用正确的编译器。于是，成功编译的块保存在S7程序下的块文件夹中。
   编译期间检测到的语法错误显示在对话框中，必须将其更正以便那些存在错误的块也能被创建。

13.7 STL源文件的实例

13.7.1 在STL源文件中声明变量的实例

基本数据类型的变量

```
              // 通过双斜杆将注释与声明部分分隔开。
VAR_INPUT     // 用于输入变量的关键字
              in1 :INT; // 变量名和变量类型用“:”分隔
              in3 :DWORD;     // 每个变量声明都以分号终止
              in2 :INT := 10;     // 用于声明中初始值的可选设置
END_VAR       // 相同声明类型的变量的声明结束
VAR_OUTPUT    // 用于输出变量的关键字
              out1 :WORD;
END_VAR       // 用于临时变量的关键字
VAR_TEMP
              temp1 :INT;
END_VAR
```

数组型数据类型的变量

```
VAR_INPUT     // 输入变量
              array1 :ARRAY [1..20] of INT;        // array1是单维数组
              array2 :ARRAY [1..20, 1..40] of DWORD;     // array2是两维数组
END_VAR
```

结构型数据类型的变量

```
VAR_OUT       //输出变量
OUTPUT1:      STRUCT// OUTPUT1具有STRUCT数据类型
              var1 :BOOL;       // 结构的元素1
              var2 :DWORD;    // 结构的元素2
              END_STRUCT;   // 结构结束
END_VAR
```

13.7.2 STL源文件中组织块的实例

```
ORGANIZATION_BLOCK OB1
TITLE = 带有不同块调用的OB1的实例
// 3个程序段显示了块调用
// 带和不带参数

{S7_pdiag := 'true'}                    // 用于块的系统属性
AUTHOR              西门子公司
FAMILY              实例
NAME                Test_OB
VERSION             1.1
VAR_TEMP
Interim value :INT;                     // 缓冲区
END_VAR

BEGIN

NETWORK
TITLE = 功能调用传送参数
// 在一行中进行参数传送
CALL FC1(param1 :=I0.0,param2 :=I0.1);


NETWORK
TITLE = 功能块调用
// 传送参数
// 在多行中进行参数传送
CALL Traffic light control , DB6(       // FB的名称、背景数据块
dur_g_p             := S5T#10S,         // 将实际值分配给参数

del_r_p             := S5T#30S,
starter             := TRUE,
t_dur_y_car         := T 2,
t_dur_g_ped         := T 3,
t_delay_y_car       := T 4,
t_dur_r_car         := T 5,
t_next_red_car      := T 6,
r_car               := "re_main",       // 引号显示了在
y_car               := "ye_main",       // 符号表中输入的符号名
g_car               := "gr_main",
r_ped               := "re_int",
g_ped               := "gr_int");


NETWORK
TITLE = 功能块调用
// 传送参数
// 在一行中进行参数传送
CALL FB10, DB100(para1 :=I0.0,para2 :=I0.1);

END_ORGANIZATION_BLOCK
```

13.7.3 STL源文件中功能的实例

```
FUNCTION FC1:VOID
//只是由于调用
VAR_INPUT
param1 :bool;
param2 :bool;
END_VAR
begin
end_function



FUNCTION FC2 :INT
TITLE = 条目的递增数
// 只要所传送的值 < 1000，该功能
// 就增加所传送的值。如果条目数
// 超过1000，将通过功能(RET_VAL)的
// 返回值返回“-1”。

AUTHOR                  Siemens
FAMILY                  Throughput check
NAME          :         INCR_ITEM_NOS
VERSION       :         1.0

VAR_IN_OUT
ITEM_NOS :INT;                  // 当前制造的条目的数目
END_VAR

BEGIN

NETWORK
TITLE = 条目以1递增
// 只要当前条目数低于1000，
// 计数器可以以1递增
L ITEM_NOS; L 1000;                     // 在一行中具有
> I; JC ERR;                            // 多个语句的实例。
L 0; T RET_VAL;
L ITEM_NOS; INC 1; T ITEM_NOS; BEU;
ERR:L -1;
T RET_VAL;
END_FUNCTION

FUNCTION FC3 {S7_pdiag := 'true'} :INT
TITLE = 条目的递增数
// 只要所传送的值 < 1000，该功能
// 就增加所传送的值。如果条目数
// 超过1000，将通过功能(RET_VAL)的
// 返回值返回“-1”。
//
// RET_VAL在此具有参数的系统属性
```

```
AUTHOR          :       Siemens
FAMILY          :       全面检查
NAME            :       INCR_ITEM_NOS
VERSION         :       1.0

VAR_IN_OUT
ITEM_NOS {S7_visible := 'true'}:INT;    // 当前制造的条目的数目
// 用于参数的系统属性
END_VAR

BEGIN

NETWORK
TITLE = 条目以1递增
// 只要当前条目数低于1000，
// 计数器可以以1递增
L ITEM_NOS; L 1000;                     // 在一行中具有
> I; JC ERR;                            // 多个语句的实例。
L 0; T RET_VAL;
L ITEM_NOS; INC 1; T ITEM_NOS; BEU;
ERR:L -1;
T RET_VAL;

END_FUNCTION
```

13.7.4 STL源文件中功能块的实例

```
FUNCTION_BLOCK FB6
TITLE = 交通灯的简单切换
// 主要街道上人行横道的
// 交通灯控制

{S7_m_c := 'true'}        // 用于块的系统属性
AUTHOR          :         Siemens
FAMILY          :         交通灯
NAME            :         Traffic light01
VERSION         :         1.3

VAR_INPUT

starter         :         BOOL    :=      FALSE; // 来自行人的过街请求
t_dur_y_car     :         TIMER;          // 用于行人通行的绿灯持续时间
t_next_r_car    :         TIMER;          // 用于车辆的红灯之间的持续时间
t_dur_r_car     :         TIMER;
number          {S7_server := 'alarm_archiv'; S7_a_type := 'alarm_8'} :DWORD;
// 车辆数目
// 数目具有参数的系统属性

END_VAR
VAR_OUTPUT

g_car           :         BOOL            :=      FALSE; // 为车辆采用GREEN

END_VAR
VAR
condition       :         BOOL    :=      FALSE; // 为车辆采用的红灯条件
END_VAR

BEGIN
NETWORK
TITLE = 用于主要街道交通的红灯条件
// 在持续至少一分钟后，人行横道处对
// 绿灯的请求构成了主要街道交通
// 的红灯条件。
                A(;
                A         #starter;       // 人行横道处的绿灯请求和
                A         #t_next_r_car;  // 红灯之间持续的时间
                选项      #condition;     // 或者红灯的条件
                );
                AN        #t_dur_y_car;   // 当前没有红灯
                =         #condition;     // 红灯条件
```

```
NETWORK
TITLE = 用于主要街道交通的绿灯
                    AN        #condition;       // 主要街道交通无红灯条件
                    =         #g_car;           // 为主要街道交通采用绿灯
NETWORK
TITLE = 用于车辆的黄灯持续时间
                    // 控制交通灯所需要的
                    // 其它程序

END_FUNCTION_BLOCK

FUNCTION_BLOCK FB10
VAR_INPUT
para1 :bool;
para2:bool;
end_var
begin
end_function_block

data_block db10
FB10
begin
end_data_block

data_block db6
FB6
begin
end_data_block
```

13.7.5 STL源文件中数据块的实例

数据块：

```
DATA_BLOCK DB10
TITLE = DB实例10
STRUCT
                    aa :BOOL;         // BOOL型变量aa
                    bb :INT;  // INT型变量bb
                    cc :WORD;
END_STRUCT;
BEGIN               //初始值的分配
                    aa := TRUE;
                    bb := 1500;
END_DATA_BLOCK
```

具有相关用户自定义数据类型的数据块：

```
DATA_BLOCK DB20
TITLE = DB(UDT)实例
UDT 20                          // 相关的用户自定义的数据类型
BEGIN
                start := TRUE;              //初始值的分配
                setp. := 10;
END_DATA_BLOCK
```

注意

所使用的UDT必须位于源文件中的数据块之前。

带有相关功能块的数据块：

```
DATA_BLOCK DB30
TITLE = DB(FB)实例
FB30                // 相关的功能块
BEGIN
                start := TRUE;      //初始值的分配
                setp. := 10;
END_DATA_BLOCK
```

注意

相关的功能块必须位于源文件中的数据块之前。

13.7.6 STL源文件中自定义数据类型的实例

```
TYPE  UDT20
STRUCT
                start :BOOL;                // BOOL型变量
                setp.: INT;                 // INT型变量
                value :WORD;                // WORD型变量
END_STRUCT;
END_TYPE
```

14 显示引用数据

14.1 可用引用数据的概述

您可创建引用数据并为其赋值，以便使用用户程序的调试和修改更容易。在下列情况下可使用引用数据：

- 作为整个用户程序的一个概述
- 作为进行修改和测试的基础
- 为了补充程序文档

下表给出了您可从各个视图中摘取的信息：:

视图	用途
交叉引用表	用户程序中所使用的存储器区I、Q、M、P、T、C，以及DB、FB、FC、SFB、SFC调用中的地址概况。 使用菜单命令**视图 > 地址的交叉引用**，可显示包括所选地址的多重访问在内的所有交叉引用。
输入、输出、和位存储器的分配表	用户程序内已占用的存储器区I、Q和M中的地址位以及定时器和计数器(T和C)等的概况；它们构成了用户程序中的故障诊断或修改的一个重要基础
程序结构	用户程序内的块的调用体系以及所使用块及其嵌套等级的概况
未用符号	为其提供有引用数据、已在符号表中定义但尚未在部分用户程序中使用的所有符号的概况
无符号的地址	为其提供有引用数据但在符号表中没有为其定义任何符号的、已在部分用户程序中使用的所有绝对地址的概况

所选用户程序的引用数据包括了表中的所有列表。也可以为一个用户程序或为一个以上的用户程序创建和显示一个或多个列表。

同时显示多个视图

例如，在附加窗口中显示其它的列表将使您能够：

- 比较不同 S7 用户程序的同一列表。
- 显示列表的各种不同视图，例如交叉引用列表，按不同方式进行显示并紧挨着放置在屏幕上。例如，您可用一个交叉引用列表只显示某一 S7 用户程序的输入而另一个列表只显示它的输出。
- 同时打开一个 S7 用户程序的多个列表，例如，程序结构和交叉引用列表。

14.1.1 交叉引用表

交叉引用表提供在 S7 用户程序中地址使用的概述。

当显示交叉引用表时，将获得存储器区输入(I)、输出(Q)、位存储区(M)、定时器(T)、计数器(C)、功能块(FB)、功能(FC)、系统功能块(SFB)、系统功能(SFC)、I/O (P)和数据块(DB)的地址列表，如同在 S7 用户程序中连同它们的地址(绝对地址或符号)和用途一起使用。它显示在激活的窗口。工作窗口的标题栏显示交叉引用表所属的用户程序的名称。

窗口的每一行对应于交叉引用表的一个条目。使用搜索功能可以方便地查找特定的地址和符号。

当显示参考数据时，交叉引用表是缺省视图。可以改变此缺省设置。

结构

交叉引用表条目由下列栏组成：

列	目录/意义
地址	绝对地址
块	使用其地址的块
类型	是否涉及地址的读取(R)和/或写(W)访问
语言/详细资料	关于创建块所使用的编程语言的信息

只有当交叉引用表选择相应的属性时，才显示块、类型、语言和详细资料栏。此块信息因写入块的编程语言而异。

可以使用鼠标根据需要设置画面中交叉引用表的栏宽度。

排序

交叉引用表缺省为存储器区排序。如果用鼠标点击栏标题，可以以缺省排序标准对条目进行排序。

交叉引用表布局的实例

地址	符号	块	类型	语言	详细信息
I1.0	电机开	OB2	R	STL	Nw 2 Inst 33 /0
M1.2	存储位	FC2	R	LAD	Nw 33
C2	计数器2	FB2		FBD	Nw2

14.1.2 程序结构

程序结构描述了 S7 用户程序中块的调用体系。同时，也给出了使用的块、它们的从属性和它们的局部数据要求的概述。

在“生成参考数据”窗口中使用菜单命令**视图 > 过滤器**，可以打开带标签的对话框。在“程序结构”标签页中，可以设置如何显示程序结构。

可以在以下两者之间选择：

- 调用结构和
- 从属性结构

程序结构的符号

符号　　含义

通常调用的块(调用FB10)

无条件调用的块(UC FB10)

有条件调用的块(CC FB10)

数据块

递归

递归和有条件的调用

递归和无条件的调用

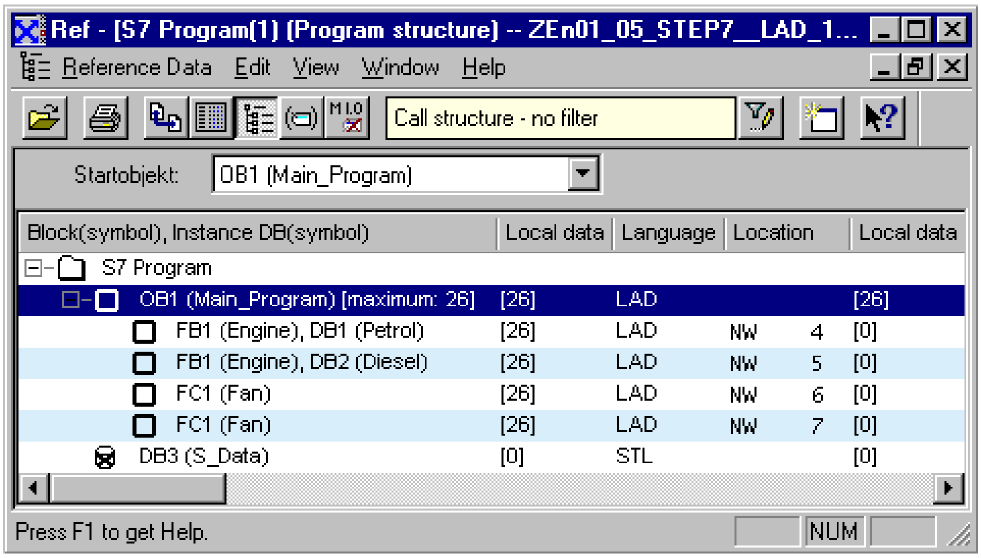
未被调用的块

- 调用中的递归在调用结构中以图形识别和显示。
- 调用体系中的递归以不同的符号显示。

- 定期调用的块(CALL)、有条件调用的块(CC)或无条件调用的块(UC)用不同的符号标记。
- 未被调用的块显示在调用结构的底部，并用黑色叉标记。对于没有被调用块在调用结构中将不再进一步细分。

调用结构

显示完整的调用体系。



如果程序结构将为所有组织块(OB)创建，而 OB1 不在 S7 用户程序中，或者如果指定的启动块不存在于程序中，将自动提示您为程序结构的根指定另一个块。

无论是调用结构还是从属结构，都可以通过选项设置来取消激活显示块的多个调用。

在调用结构中显示最大的局部数据要求

为了总览在用户程序中显示的组织块的局部数据要求，可以用树结构显示下列信息：

- 每个 OB 的最大局部数据要求和
- 每个路径的局部数据要求

可以在“程序结构”标签中激活和取消激活此显示。

如果存在同步错误 OB (OB121、OB122)，将在最大局部数据要求的数字值后显示用于同步错误 OB 的加号和附加的要求。

从属性结构

从属性结构显示每个块在项目中与其它块的从属性。此块显示在左边外侧，在下面列出的在锯齿状的段中的是调用或使用此块的块。

显示删掉的块

与删掉的块相关的行以红色加亮显示。

14.1.3 分配列表

分配列表显示哪些地址已分配到用户程序中。此显示是故障诊断或在用户程序中进行修改的重要基础。

I/Q/M 分配列表概述了存储器区输入(I)、输出(Q)、位存取区(M)、定时器(T)和计数器(Z)的哪个字节的哪个位的使用情况。I/Q/M 分配列表显示在工作窗口中。

工作窗口的标题栏显示分配列表所属的 S7 用户程序的名称。

I/Q/M 表

每一行包含存储器区的一个字节，该字节的八个位根据它们的访问进行编码。它也指出是字节、字还是双字的访问。

I/Q/M 表中的标识

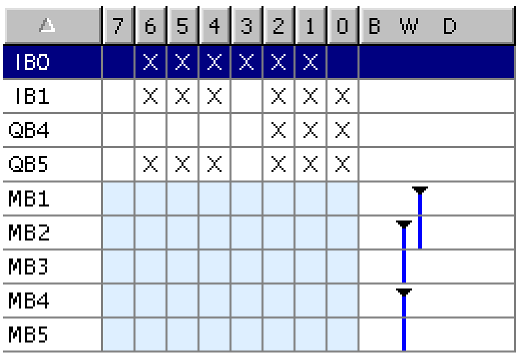
白色背景	没有访问该地址，因而对其分配。
X	直接访问地址。
蓝色背景	间接访问地址(字节、字或双字访问)。

I/Q/M表中的栏

列	目录/意义
7 6 5 4 3 2 1 0	相应字节的位编号
B	字节被一个字节访问占用
W	字节被一个字访问占用
D	字节被一个双字访问占用

实例

下面的实例给出了输入、输出和位存取区(I/Q/M)分配列表的典型布局。



	7	6	5	4	3	2	1	0	B W D
IB0		X	X	X	X	X	X		
IB1		X	X	X		X	X	X	
QB4						X	X	X	
QB5		X	X	X		X	X	X	
MB1									
MB2									
MB3									
MB4									
MB5									

第一行显示输入字节 IB 0 的分配。地址 IB 0 的输入为直接访问(位访问)。

栏“0”、“1”、“2”、“3”、“5”和“6”以位访问“X”标识。也有字访问存储器字节 1 和 2、2 和 3 或 4 和 5。因此，一个“横条”显示在“W”栏中，并且单元格也以淡蓝色背景显示。横条的黑色末端显示字访问的开始。

T/C 表

每行显示 10 个定时器或计数器。

实例

	0	1	2	3	4	5	6	7	8	9
T 00-09	.	T1	.	.	.		T6	.	.	.
T 10-19	.	.	T12	.	.	.	.	T17	.	T19
T 20-29	.	.	.	.	T24	.	.	.	.	.
Z 00-09	.	.	Z2	.	.	.	.	Z7	.	.
Z 10-19	.	.	.	.	.	.	.	.	.	Z19
Z 20-29	.	.	.	.	.	.	.	.	.	.
Z 30-39	.	.	.	.	Z34	.	.	.	.	.

14.1.4 未使用的符号

显示具有下列特征的所有符号的概述：

- 在符号表中定义的符号。
- 符号没有在存在参考数据的用户程序的部分中使用。

它们显示在激活的窗口中。工作窗口的标题栏显示列表所属的用户程序的名称。

窗口的每一行对应于列表的一个条目。每行包括地址、符号、数据类型和注释。

列	目录/意义
地址	绝对地址
数据类型	地址的数据类型
注释	关于来自符号表中的地址的注释

未使用的符号布局的列表实例

符号	地址	数据类型	注释
MCB1	I 103.6	BOOL	电机电路断路器1
MCB2	I 120.5	BOOL	电机电路断路器2
MCB3	I 121.3	BOOL	电机电路断路器3

可以通过点击栏标题对条目排序。

14.1.5 没有符号的地址

当显示没有符号的地址列表时，将获得在S7用户程序中使用的元素列表，而这些元素没有在符号表中定义。它们显示在激活的窗口中。工作窗口的标题栏显示列表所属的用户程序的名称。

行包括地址以及地址在用户程序中使用的定时器的编号。

实例：

地址	编号
Q 2.5	4
I 23.6	3
M 34.1	20

列表根据地址排序。

14.1.6 为LAD、FBD和STL显示块信息

梯形逻辑、功能方框图和语句表的语言相关信息显示在交叉引用表和程序结构中。此信息包括块语言和它的详细资料。

如果过滤器在“程序结构”标签中设置为“调用结构”，并且如果选择相应的选项，“程序结构”视图只显示语言相关的信息。

“交叉引用”中的语言相关信息可以通过菜单命令**视图 > 过滤器**显示或隐藏。

- 在“过滤器”对话框的“交叉引用”标签页中激活“块语言”和“详细资料”复选框，以显示块语言信息。

语言相关信息因块写入的编程语言而异，并且用缩写显示。

语言	网络	语句	指令
STL	Nw	Inst	/
LAD	Nw		
FBD	Nw		

Nw和**Inst**指定在哪一个程序段和哪一个语句中使用了地址(交叉引用表)或调用了块(程序结构)。

为可选择的编程语言显示块信息

如果安装相应的可选程序包，就可以访问关于块信息的在线帮助主题。

14.2 处理引用数据

14.2.1 显示参考数据的方法

下列方法可能会用来显示参考数据：

从 SIMATIC 管理器显示

1. 在项目窗口的离线组件视图中，选择“块”文件夹。
2. 选择菜单命令**选项 > 引用数据 > 显示**。

从编辑器窗口显示

1. 在“块”文件夹中打开一个块。
2. 在编程语言编辑器窗口中，选择菜单命令**选项 > 参考数据**。

显示“自定义”对话框。在此可选择最先显示的视图。缺省视图为在应用中最后关闭的用于显示参考数据的视图。可隐藏该对话框，用于将来调用。

直接从已编译的块中显示

可以直接从语言编辑器中显示已编译块的参考数据，获取用户程序的当前概况。

14.2.2 在附加的工作窗口显示列表

使用菜单命令**窗口 > 新建窗口**可以打开附加工作窗口和显示其他参考数据的视图(例如，未使用的符号列表)。

使用菜单命令**参考数据 > 打开**可打开一个工作窗口以显示先前隐藏的参考数据。

通过选择“视图”菜单中的命令或选择工具栏上相应的按钮，可以切换到其它参考数据的视图：

参考数据视图	显示此参考数据视图的菜单命令
没有符号的地址	视图 > 没有符号的地址
未使用的符号	视图 > 未使用的符号
分配	视图 > 分配
程序结构	视图 > 程序结构
交叉引用表	视图 > 交叉引用

14.2.3 生成和显示引用数据

生成参考数据：

1. 在SIMATIC管理器中，选择希望生成参考数据的块文件夹。
2. 在“SIMATIC管理器”中选择菜单命令**选项 > 参考数据 > 生成**。

在生成参考数据前，计算机检查是否有任何可用的参考数据，如果有，则检查数据是否是当前的。

- 如果参考数据是可用的，它们将被生成。
- 如果可选的的参考数据不是当前的，可以选择是否刷新参考数据或是否再次完全生成它们。

显示参考数据：

使用菜单命令选项 > 参考数据 > 显示可以显示参考数据。

在显示参考数据前，进行检查以确定是否存在参考数据，以及存在的参考数据是否是当前的。

- 如果不存在参考数据，则将生成它们。
- 如果存在不完整的参考数据，将显示一个对话框，提醒参考数据不一致。然后可以决定是否要刷新参考数据以及到什么程度。有下列可能性：

选项	含义
仅适用于修改过的块	刷新任何修改或新建的块的参考数据；任何删掉的块的信息将从参考数据库中移除。
对所有的块	完全重新生成所有的块参考数据。
不刷新	不刷新参考数据。

为了刷新参考数据，将对块重新编译。调用合适的编译器以编译每个块。使用菜单命令**视图 > 刷新**可以刷新已在激活窗口显示的参考数据的视图。

14.2.4 在程序中快速搜索地址位置

在编程时可以使用参考数据将光标放置到程序中地址的不同位置。为此，必须有最新的参考数据。然而，不必启动应用程序来显示参考数据。

基本步骤

1. 在SIMATIC管理器中选择菜单命令**选项 > 参考数据 > 生成**以生成当前的参考数据。只有当没有参考数据或只有旧的参考数据时，才需要此步骤。
2. 在打开的块中选择地址。
3. 选择菜单命令**编辑 > 跳转到 > 实例**。
随后显示一个对话框，其中包含程序中所有地址实例的列表。
4. 如果还要显示其物理地址或地址区域与调用的地址重叠的地址实例，选择选项“重叠访问存储器区”。于是“地址”栏增加到表中。
5. 在列表中选择位置，并点击“跳转到”按钮。

当打开对话框时如果参考数据不是最新的，将显示此结果的信息。然后，可以刷新参考数据。

位置列表

对话框中的位置列表包含下列详细资料：

- 地址被使用的块
- 块的符号名(如果存在的话)
- 详细资料(例如关于位置的信息)，以及指令(如果合适的话。它们取决于块或源文件(SCL)的原始编程语言)
- 语言相关的信息
- 地址的访问类型：只读(R)、只写(W)、读写(RW)、未知(?)。
- 块语言

可以过滤位置的显示，例如，可以只查看对一个地址的写访问。关于在域中输入什么和其它显示的信息，此对话框的在线帮助提供了更详细的信息。

注意

参考数据仅离线存在。因此，该功能总是用于离线块的交叉引用，即使它是被一个在线块调用。

14.2.5 使用地址位置的示例

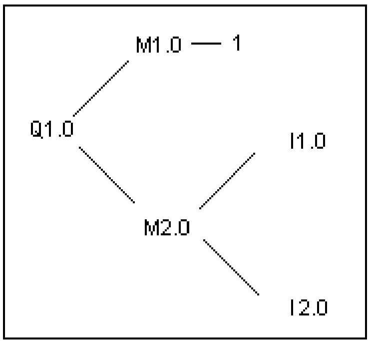
希望确定在哪个位置设置输出 Q1.0 (直接/间接)。下列在 OB1 中的 STL 代码用作示例：

Network 1: ....
A Q 1.0 // 与本例
= Q 1.1 //在此示例中

Network 2：
A M1.0
A M2.0
= Q 1.0 // 赋值

Network 3：
//仅适用于注释行
SET
= M1.0 // 赋值

Network 4：
A I 1.0
A I 2.0
= M2.0 // 赋值
其结果如 Q1.0 的下列分配树所示：



然后如下进行操作：

1. 在LAD/STL/FBD编辑器中，将光标置于OB1的Q1.0 (NW 1, Inst 1)上。
2. 选择菜单命令**编辑 > 跳转到 > 位置**或使用鼠标右键选择“跳转到位置”。
   对话框将显示Q1.0的所有分配：
   OB1 循环执行 NW 2 Inst 3 /= W STL
   OB1 循环执行 NW 1 Inst 1 /A R STL
3. 使用对话框中的“跳转到”按钮在编辑器中跳转到“NW 2 Inst 3”：
   Network 2:
   A M1.0
   A M2.0
   = Q 1.0
4. 现在必须检查M1.0和M2.0的分配。首先将光标置于LAD/STL/FBD编辑器中的M1.0上。

5. 选择菜单命令**编辑 > 跳转到 > 位置**或使用鼠标右键选择“跳转到位置”。对话框将显示M1.0的所有分配：
   OB1 循环执行 NW 3 Inst 2 /= W STL
   OB1 循环执行 NW 2 Inst 1 /A R STL
6. 在对话框中使用“跳转到”按钮，跳转到编辑器中的“NW 3 Inst 2”。
7. 在LAD/STL/FBD编辑器的程序段3中，将看到对M1.0的分配并不重要(因为它始终是TRUE)，相反对M2.0的分配需要检查。

   在早于V5的STEP 7版本中，将必须再次全部运行整个分配序列。按钮“>>”和“<<”使之相当简易：

8. 将打开的对话框“跳转到位置”放在顶部，或在LAD/STL/FBD编辑器中从当前位置调用功能“跳转到位置”。
9. 点击“<<”按钮一次或两次直到显示所有的Q1.0位置；最后的跳转位置“NW 2 Inst 3”将被选择。
10. 使用“跳转到”按钮(如同第3点)，从地址位置对话框跳转到“NW 2 Inst 3”：
    Network 2:
    A M1.0
    A M2.0
    = Q 1.0
11. 在第4点中，已检查了M1.0的分配。现在必须检查所有(直接/间接)对M2.0的分配。将光标放在编辑器中的M2.0上，并调用功能“跳转到位置”：将显示所有对M2.0的分配：
    OB1 循环执行 NW 4 Inst 3 /= W STL
    OB1 循环执行 NW 2 Inst 2 /A R STL
12. 使用“跳转到”按钮跳转到LAD/STL/FBD编辑器中的“NW 4 Inst 3”：
    Network 4:
    A I 1.0
    A I 2.0
    = M2.0
13. 现在必须检查I1.0和I2.0的分配。本例不描述此过程，因为它的处理与前面的方法相同(前4点)。

通过在 LAD/STL/FBD 编辑器和地址位置对话框之间切换，可以搜索和检查程序中相关的位置。

15 选中“块一致性和时间标记”作为块属性

15.1 检查块一致性

介绍

如果各个对象的接口或代码必须调整或扩展，就可能导致时间标记冲突。时间标记冲突反过来可以引起调用对象和被调用对象或引用块之间的块不一致，从而导致大量的更正工作。

“检查块一致性”功能消除了许多这类更正工作。“检查块一致性”功能会删除全部时间标记冲突和块不一致性的大部分。在对象的块不一致性不能自动消除的情况下，该功能会在相应编辑器中找到要改变的地方，您可在该处进行所需要的改变。所有块不一致性的消除和对象的编译都是逐步完成的。

要求

只能检查用STEP 7 V5.0，Service Pack 3以上版本创建的项目的块一致性。对于更旧的项目，当开始块一致性检查时，必须首先全面编译(菜单命令**程序 > 全面编译**)。

对于用选项包创建的对象，必须为一致性检查安装选项包。

开始块一致性检查

在开始块一致性检查时，会检查块接口的时间标记，可能引起块不一致的对象在树形视图中高亮度显示(相关树：引用/调用树)。

1. 在SIMATIC管理器中，跳转到项目窗口，选择所需要的块文件夹，然后通过菜单命令**编辑 > 检查块一致性**启动块一致性检查。

2. 在“检查块一致性”中选择菜单命令**程序 > 编译**
   STEP 7自动识别相关对象的编程语言，并调用相关的编辑器。尽可能自动更正时间标记冲突和块不一致性，并且编译对象。如果对象中的时间标记冲突或不一致性不能自动消除，则输出窗口中会显示错误消息(更多步骤参见第3步)。对树形视图中的所有对象自动重复该过程。
3. 如果在编辑运行期间，不可能自动消除所有的块不一致性，则相应对象会在输出窗口中作为错误消息标记。将鼠标放在相应的错误条目上，并使用鼠标右键在弹出式菜单中调用错误显示。相关的错误打开，程序跳转到要改变的位置。消除所有的块不一致性，保存并关闭对象。对所有标记为错误的对象重复该过程。
4. 再次开始步骤2和步骤3。重复该过程，直到消息窗口中不再有错误显示。

15.2 时间标记作为块属性和时间标记冲突

块包含代码时间标记和接口时间标记。这些时间标记显示在块属性的对话框中。可以使用时间标记监视 STEP 7 程序的一致性。

如果比较时间标记时检测到违反规则，STEP 7 就显示时间标记冲突。可能会发生下列违规：

- 被调用的块比调用块(调用)更新。
- 被引用的块比使用它的块更新。
- 第二类违规的实例：
- UDT 比使用它的块更新；即，使用变量声明表中的 UDT 的 DB 或另一个 UDT、或 FC、FB 或 OB。
- FB 比其相关的背景数据块更新。
- FB2 被定义为 FB1 中的多个实例，而 FB2 比 FB1 更新。

注意

即使接口时间标记之间的关系是正确的，也可能发生不一致性：

- 引用块接口的定义与其使用位置处的定义不匹配。

这些不一致性被认为是接口冲突。例如，当块从不同程序复制，或当编译ASCII源文件而未生成程序中所有的块时，就会发生。

15.3 逻辑块中的时间标记

代码时间标记

在此可输入创建块的时间与日期。对时间标记进行更新：

- 在程序代码已修改时
- 在接口描述已修改时
- 在注释已修改时
- 在首次创建 ASCII 源文件并进行编译时
- 在块属性(“属性”对话框)已修改时

接口时间标记

对时间标记进行更新：

- 当接口描述已修改时(数据类型或初始值发生变化，新的参数)
- 当首次创建 ASCII 源文件并进行编译时，如果从结构上修改了接口。
- 不对时间标记进行更新：
- 当符号发生变化时
- 当变量表中的注释发生变化时
- 当 TEMP 区中进行了改动时

块的调用规则

- 已调用块的接口时间标记必须比正调用块的代码时间标记要早。
- 如果没有打开任何调用该块的块，则只能改变该块的接口。否则，如果保存比已修改块时间更迟的正调用块，那么，将无法从时间标记中识别这种冲突。

发生时间标记冲突时的处理过程

当打开一个正在调用的块时，将显示时间标记冲突。在对 FC 或 FB 接口进行修改之后，在调用块时对该块的所有调用都将按扩展形式显示。

如果块的接口发生变化，则调用该块的所有块也必须进行调整。

在对 FB 接口进行修改之后，现有的多重背景定义和背景数据块都必须进行更新。

15.4 共享数据块中的时间标记

代码时间标记

对时间标记进行更新：

- 在首次创建 ASCII 源文件时
- 在编译 ASCII 源文件时
- 当在对块的声明视图或数据视图中进行了改动时

接口时间标记

对时间标记进行更新：

- 当声明视图中的接口描述发生了变化时(数据类型或初始值发生变化，新的参数)

15.5 背景数据块中的时间标记

背景数据块将保存功能块的形式参数和静态数据。。

代码时间标记

在此可输入背景数据块创建时的时间和日期。当在背景数据块的数据视图中输入实际值时，将对时间标记进行更新。用户不能对背景数据块的结构进行修改，因为结构来源于相关的功能块(FB)或系统功能块(SFB)。

接口时间标记

当创建背景数据块时，将输入相关 FB 或 SFB 的接口时间标记。

无冲突的打开规则

FB/SFB 的接口时间标记与其相关背景数据块必须匹配。

发生时间标记冲突时的处理过程

如果修改了FB的接口，那么，将对FB的接口时间标记进行更新。当打开相关的背景数据块时，将报告出现了一个时间标记冲突，因为背景数据块与FB的时间标记不再匹配。在数据块的声明段中，接口将用由编译器所生成的符号来显示(伪符号)。背景数据块现在只能进行浏览。

为纠正这种类型的时间标记冲突，必须重新为已修改的 FB 创建背景数据块。

15.6 UDT中以及来源于UDT的数据块中的时间标记

用户自定义的数据类型(UDT)可用于创建具有同一结构的许多数据块。

代码时间标记

每次进行修改时都要对代码时间标记进行更新。

接口时间标记

当接口描述发生变化时，对接口时间标记进行更新(数据类型或初始值发生变化，新的参数)。

在对 ASCII 源文件进行编译时，UDT 的接口时间标记也将进行更新。

无冲突的打开规则

- 用户自定义数据类型的接口时间标记，必须比其中使用了该数据类型的逻辑块的接口时间标记要早。
- 用户自定义数据类型的接口时间标记，必须与来源于 UDT 的数据块的时间标记完全相同。
- 用户自定义数据类型的接口时间标记，必须比下一级 UDT 的时间标记要新。

发生时间标记冲突时的处理过程

如果改变了数据块、功能块、或另一个 UDT 中所使用的 UDT 定义，那么，当块打开时，STEP 7 将报告出现了时间标记冲突。

UDT 组件将显示为一个分散的结构。所有变量名称均将被系统预先设置的值所覆盖。

15.7 对功能、功能块、或UDT中的接口进行纠正

如果需要对 FB、FC、或 UDT 中的接口进行纠正，为避免出现时间标记冲突，可如下进行操作：

1. 从想要改变的块中生成一个STL源文件，并生成所有直接或间接引用的块。
2. 将变化保存在您生成的源文件中。
3. 将已修改的源文件重新编译到块中。

现在即可保存/下载接口变化。

15.8 避免调用块时出现错误

STEP 7 覆盖 DB 寄存器中的数据

当执行各种不同的指令时，STEP 7 将修改 S7-300/S7400 CPU 的寄存器。例如，当调用 FB 时，DB 和 DI 寄存器的内容将互换。这将使得打开所调用 FB 的背景数据块时不会丢失前一个背景数据块的地址。

如果使用绝对地址进行操作，那么，在访问保存在寄存器中的数据时可能出现错误。在某些情况下，寄存器 AR1 (地址寄存器 1)以及 DB 寄存器中的地址均将被覆盖。这意味着可能读取或写入错误的地址。

危险

在下列情况下可能存在对人员和财产造成损坏的危险：

1. 使用CALL FC、CALL FB、CALL多重背景
2. 使用完整的绝对地址访问DB (例如DB20.DBW10)
3. 访问复杂数据类型的变量

有可能DB寄存器(DB和DI)、地址寄存器(AR1、AR2)、以及累加器(ACCU1、ACCU2)的内容都发生变化。

此外，当调用FB或FC时，可能无法使用作为附加(隐含)参数的状态字的RLO位。

当使用上述的编程方法时，必须确保自己保存和恢复内容；否则，可能发生错误。

保存正确数据

如果使用缩写格式访问数据的绝对地址，则 DB 寄存器的内容可能导致出现危险情况。例如，如果假定 DB20 已打开(也就是说其编号已保存在 DB 寄存器中)，那么，可指定 DBX0.2 来访问 DB 的 0 字节的第 2 位的数据，DB 的地址已输入到 DB 寄存器中(换句话说，就是 DB20)。然而，如果 DB 寄存器包含有一个不同的 DB 号，那么，将访问到错误的数据。

通过使用下列方法对数据进行寻址，可避免在访问 DB 寄存器的数据时出现错误：

- 使用符号地址
- 使用完整的绝对地址(例如 *DB20.DBX0.2*)

如果使用这些寻址方法，STEP 7 将自动打开正确的 DB。如果使用用于间接寻址的 AR1 寄存器，必须始终将正确的地址装载到 ARI 中。

对寄存器进行修改的情形

只有在 STL 中对用于间接寻址的地址寄存器进行操作才是合乎逻辑的。其它编程语言均不支持对地址寄存器的间接访问。

在所有的编程语言中都必须考虑编译器对 DB 寄存器的调整，以便确保调用块时传送正确的参数。

所调用块的地址寄存器 AR1 和 DB 寄存器的内容在下列情形下均将被覆盖：

情形	描述
带有来自DB的实际参数	• 一旦将实际参数分配给来自DB的块(例如DB20.DBX0.2)，STEP 7就将打开DB (DB20)，并调整DB寄存器的内容。程序随后在块调用之后将使用经调整的DB进行工作。
在调用具有更高级数据类型的块时	• 当块已经从FC内进行了调用，该FC将把具有更高数据类型(字符串、数组、结构或UDT)的形式参数的内容传送给所调用的块，在此之后，所调用块的AR1和DB寄存器的内容均将被修改。 • 如果参数位于调用者的VAR_IN_OUT区，则也适用于FB内的调用。
当访问具有更高级数据类型的组件时	• 当FB访问VAR_IN_OUT区中具有更高级数据类型(字符串、数组、结构或UDT)的形式参数的组件时，STEP 7将使用地址寄存器AR1和DB寄存器。这意味着将修改两个寄存器中的内容。 • 当FC访问VAR_IN_OUT区中具有更高级数据类型(字符串、数组、结构或UDT)的形式参数的组件时，STEP 7将使用地址寄存器AR1和DB寄存器。这意味着将修改两个寄存器中的内容。

注意

- 当在版本1的块中调用FB时，如果在调用之前命令没有限制RLO，那么将不能正确地传送用于第一个布尔型IN或IN_OUT参数的实际参数。在这种情况下，它将在逻辑上与现有的RLO进行组合。
- 当调用FB时(单个或多重背景)，将写入地址寄存器AR2。
- 如果在FB中修改了地址寄存器AR2，那么，将不能保证正确执行FB。
- 如果没有将完整的绝对DB地址传送给ANY参数，ANY指针将不能获取打开的DB的DB号。取而代之，它将总是得到编号0。

---

16 组态消息

16.1 消息概念

消息允许您在可编程控制器工作期间，快速检测、定位和修正错误，从而显著降低设备的停工时间。

在输出消息之前，必须首先组态消息。

通过 STEP 7，可以利用已分配的消息文本和消息属性来创建、编辑与事件链接的消息。还可以编译消息，然后在显示设备上进行显示。

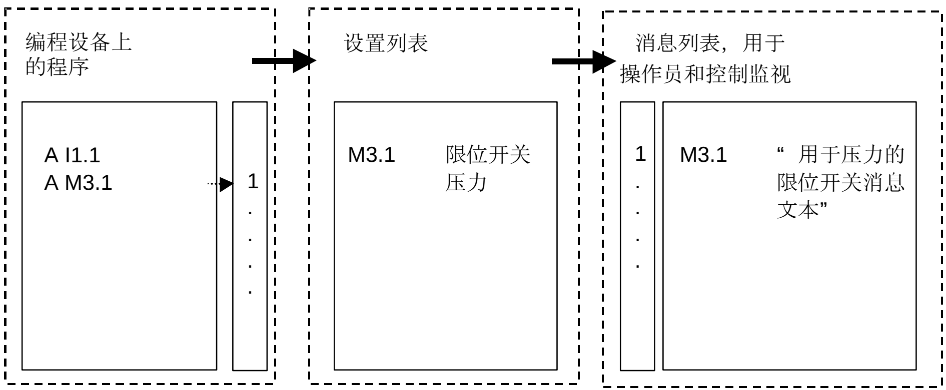
16.1.1 有哪些不同的消息传送方法？

有不同的消息创建方法。

位消息传送

位消息传送要求程序员执行三个步骤：

- 在编程设备上创建用户程序，然后设置所要求的位。
- 使用任意一个文本编辑器来创建分配列表，在该编辑器中将消息文本分配给消息位(例如，M 3.1 = 限位开关电压)。
- 根据分配列表，在操作面板上创建消息文本列表。

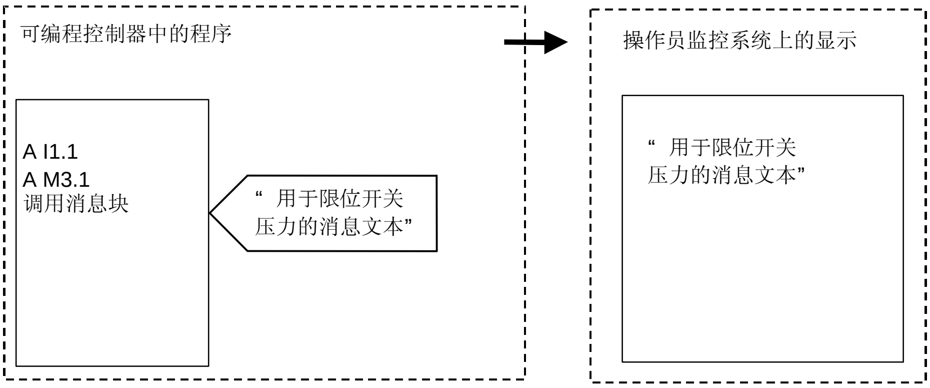


操作员接口系统会循环查询可编程控制器，查看消息位是否发生变化。如果可编程控制器的消息位发生变化，那么显示相应的消息。该消息接收来自操作员接口系统的时标。

消息号

消息号只要求程序员执行一个步骤：

- 编程期间，在编程设备上创建用户程序，设置所要求的位，然后将所要求的消息文本直接分配给位。



可编程控制器不进行循环查询。当可编程控制器信号发生变化时，相应的消息号就传送到操作员接口系统，然后显示相应的文本消息。消息接收来自可编程控制器的时标，因此，其跟踪比位消息传送更精确。

16.1.2 选择一种消息传送方法

概述

下表显示了不同消息传送方法的属性和要求：

消息号	位消息传送
• 在编程设备和操作面板的公用数据库中管理消息 • 总线上的负载为低(可编程控制器发布激活信号) • 消息接收来自可编程控制器的时标	• 编程设备和操作面板没有公用数据库 • 总线上的负载为高(操作面板轮询) • 消息接收来自操作面板的时标

消息号方法识别下列三种类型的消息：

与块有关的消息	与符号有关的消息	用户自定义的诊断消息
• 与程序同步 • 使用WinCC和ProTool (仅适用于ALARM_S)显示 • 可用于S7-300/400 • 通过消息块编程： - ALARM - ALARM_8 - ALARM_8P - NOTIFY - NOTIFY_8P - ALARM_S(Q) - AR_SEND - ALARM_D(Q) • 传送到操作面板 - 对于WinCC，通过AS-OS链接组态 - 对于ProTool，通过ProTool功能	• 与程序异步 • 通过WinCC显示 • 只能用于S7-400 • 通过符号表组态 • 通过系统数据块(SDB)下载到PLC • 通过AS-OS链接组态传送到操作面板	• 与程序同步 • 在编程设备的诊断缓冲区中显示 • 可用于S7-300/400 • 通过消息块(系统功能)编程 - WR_USMSG • 不传送到操作面板

STEP 7 只支持用户界面更友好的消息号方法，下文将详细描述该方法。在人机界面设备中组态位消息传送，也在此进行描述。

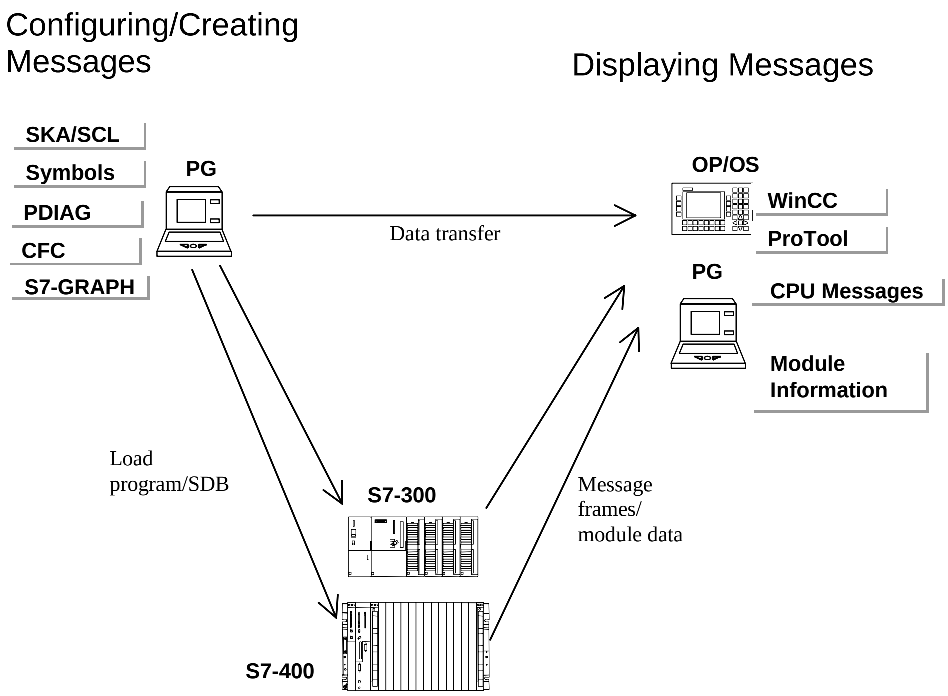
消息号方法实例

消息传送方法	应用
与块有关的消息	用于报告程序同步事件，例如，用于显示控制器已经到达限制值
与符号有关的消息	用于报告与程序无关的事件，例如，受监视的开关设置
用户自定义消息	用于报告每次调用SFC时，诊断缓冲区中的诊断事件

16.1.3 SIMATIC组件

概述

下图给出了在组态和显示消息中所涉及的 SIMATIC 组件概况。



16.1.4 消息组成

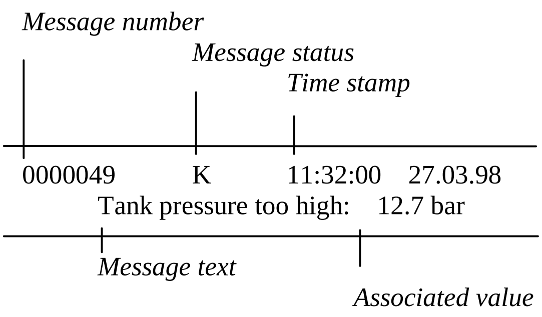
消息的显示方式取决于消息传送方法、所使用的消息块以及显示设备。

下表列出了消息的可能组成部分：

组成	描述
时标	发生消息事件时，在可编程控制器中生成
消息状态	可能为下列状态：进入、离开、无应答离开、有应答离开
关联值	可给一些消息分配由所使用的消息块评估的过程值
映像	如果系统崩溃，那么随后可在操作员站上显示所发生的消息。
消息号	项目或CPU中的唯一编号。该编号由系统分配，用于识别消息。
消息文本	由用户组态

实例

下面的实例显示了操作面板上的一个报警消息。



16.1.5 有哪些消息块可供使用？

可以在下列消息块中选择，其中每个消息块都包含一个已编程的消息功能：

- SFB 33: “ALARM“
- SFB 34: “ALARM_8“
- SFB 35 “ALARM_8P“
- SFB 36 “NOTIFY“
- SFC 18: “ALARM_S“和 SFC 17: “ALARM_SQ“
- SFB 37: “AR_SEND“(用于发送归档；不能组态消息文本和消息属性)
- SFB 31: “NOTIFY_8P“
- SFC 107: “ALARM_DQ“
- SFC 108: “ALARM_D“

可从块的在线帮助中得到详细信息。

这些消息块分别在什么情况下使用

下表有助于您为特定的任务选择消息块。根据下列事项选择消息块：

- 块中可使用的通道数目以及每个块调用监视的信号数目
- 消息是否需要应答
- 同时能指定相关值的选项
- 使用的显示设备
- 使用的 CPU 的项目数据。

消息块	通道	应答	相关值	**WinCC 显示**	**ProTool 显示**	**CPU消息 /S7状态显示**	**PLC**	说明
ALARM SFB33	1	可以	最大为 10	是	否	否	S7-400	在每个进入或离开边沿发送一个消息
ALARM_8 SFB34	8	可以	否	是	否	否	S7-400	在一个或多个信号的每个进入或离开边沿发送一个消息
ALARM_8P SFB35	8	可以	最大为 10	是	否	否	S7-400	同ALARM_8
NOTIFY SFB36	1	否	最大为 10	是	否	否	S7-400	同ALARM
NOTIFY_8P SFB 31	8	否	最大为 10	是	否	否	S7-400	同NOTIFY
AR_SEND SFB37	1			是	否	否	S7-400	用于发送一个归档；不能组态消息文本和消息属性
ALARM_SQ SFC17	1	可以	1	是	是*	是	S7-300/ S7-400	不是某个端变化、而是每次SFC调用会生成一个消息
ALARM_S SFC18	1	否	1	是	是*	是	S7-300/ S7-400	同ALARM_SQ
ALARM_DQ SFC 107	1	可以	1	是	是	是	S7-300/ 400	同ALARM_SQ
ALARM_D SFC 108	1	否	1	是	是	是	S7-300/ 400	同ALARM_SQ
*取决于OP类型								

16.1.6 形式参数、系统属性和消息块

作为消息号输入的形式参数

对于每个消息或消息组，在程序中需要一个形式参数，在程序变量概述中将其指定为IN参数。然后该形式参数用作消息号输入，构成消息基础。

如何为形式参数提供系统属性

作为启动消息组态的前提条件，首先必须按如下方式为形式参数提供系统属性：

1. 给参数添加下列系统属性：“S7_server”和“S7_a_type”
2. 将值分配给对应于程序代码中所调用的消息块的系统属性。“S7_server”的值始终为“alarm_archiv”，而“S7_a_type”的值与所调用的消息块一致。

系统属性和相应的消息块

消息块本身在消息服务器中不作为对象显示；相反，显示包含系统属性“S7_a_type”的相应值。这些值的名称与作为SFB或SFC存在的消息块名称相同(例外：“alarm_s”)。

S7_a_type	消息块	描述	属性
alarm_8	ALARM_8	SFB34	8个通道、有应答、无相关值
alarm_8p	ALARM_8P	SFB35	8个通道、有应答、每个通道最多10个相关值
notify	NOTIFY	SFB36	1个通道、无应答、最多10个相关值
alarm	ALARM	SFB33	1个通道、有应答、最多10个相关值
alarm_s	ALARM_S	SFC18	1个通道、无应答、最多1个相关值
alarm_s	ALARM_SQ	SFC17	1个通道、有应答、最多1个相关值
ar_send	AR_SEND	SFB37	用于发送一个归档
notify_8p	NOTIFY_8P	SFB 31	8个通道、无应答、最多10个相关值
alarm_s	ALARM_DQ	SFC 107	1个通道、无应答、最多1个相关值
alarm_s	ALARM_D	SFC 108	1个通道、无应答、最多1个相关值

可以在参考在线帮助中获得关于系统属性的更多详细信息。

如果程序中使用消息块的是具有相应系统属性的SFB或FB，那么自动分配系统属性，并作为多实例调用。

16.1.7 消息类型和消息

消息组态允许使用不同的过程来创建消息类型或消息。这取决于用于访问消息组态的消息类型块。

消息类型块可以是功能块(FB)或背景数据块。

- 通过 FB，可以创建一个消息类型，并可作为创建消息的一个类型。消息会自动具有该消息类型具有的所有条目。如果给功能块分配了背景数据块，那么背景数据块的消息会根据消息类型和已有的消息号自动生成。
- 对于背景数据块，基于有特定背景的消息类型生成的消息是可以修改的。

此处的显著区别是消息号是分配给消息而不是消息类型的。

锁定消息模板数据

通过消息组态可以给与事件有关的消息输入文本和属性。例如，还可以指定消息在特定显示设备上的显示方式。使用消息类型可以更方便地生成消息。

- 当为消息类型输入数据(属性和文本)时，可以选择是否锁定这些数据。属性锁定时，在输入框旁边添加一个钥匙符号，或在“锁定”栏放置一个复选标记。锁定文本在“锁定”栏中显示一个复选标记。
- 对于有“锁定数据”的消息类型，就不能修改特定背景消息。只能显示数据。
- 如果的确需要修改，那么必须返回到消息类型，删除锁定，然后进行修改。该修改不应用于修改之前生成的背景。

修改消息模板数据

对消息类型数据的修改是否会影响背景，是取决于生成项目时，是否将消息号全局分配给项目(面向项目的消息号)或 CPU (面向 CPU 的消息号)。

- 分配面向项目的消息号：希望后来的修改能应用到背景的消息类型数据时，也必须相应地在背景中修改数据。

- 将消息号分配给 CPU：消息类型数据后来的修改能自动应用到背景中。
  例外：先先前您已经在背景中修改了数据或随后对消息类型数据进行锁定或解除闭锁。如果将 FB 和背景数据块从带已分配给项目的消息号的项目复制到带已分配给 CPU 的消息号的项目时，则必须按与消息模板相同的方式改变背景的数据。

警告：

将背景复制到另一个程序且不含消息模板时，可能只能部分显示该背景。为进行补救，请将消息模板复制到新程序中。

16.1.8 如何从消息类型块中生成STL源文件

从消息文本块中生成 STL 源文件时，组态信息也将写入到源文件。

该信息写入到一个以“*$ALARM_SERVER”开头，以“*”结尾的伪注释中。

警告

设置块的符号参考时，请注意在编译源文件之前可能无法修改符号表。

当源文件包含多个块时，多个伪注释块组合形成一个注释块。不能从 STL 源文件中删除具有消息属性的单个块。

还将系统文本库复制到源文件。由于它们始终与消息关联，因此它们将作为结构集成到消息的源文件中。

16.1.9 分配消息号

可以指定是否给项目或给CPU分配消息号。给CPU分配消息号的优点在于允许复制一个程序而无需修改消息号，但是修改编号需要重新编译。只能给安装“WinCC V6.0”和/或“ProTool V6.0”应用程序的人机界面设备上的CPU显示消息号。如果使用这些应用程序的较早版本，那么必须选择项目的消息号。

16.1.10 将消息号分配给项目与分配给CPU之间的区别

下表显示了为项目和为CPU分配消息号之间的区别：

项目范围	CPU范围
一些消息属性和文本取决于所使用的HMI单元，必须组态特定显示。	分配的属性和文本不依赖于所使用的HMI单元，即，不必输入更多显示设备，或为该设备规定特定的显示消息。
程序在复制后必须重新编译。	程序可复制到项目的其它位置和到其它项目。然而，如果仅复制了单个块，则程序必须重新编译。
当随后改变消息类型数据(文本和属性)时，必须也修改实例。	如果随后改变消息类型数据(文本和属性)，所有的改变都自动应用于实例(例外：已事先改变了实例的数据)。
文本只能写在一行中。	文本可以写在多行中。

16.1.11 用于修改项目消息号分配的选项

在SIMATIC管理器的“消息号”标签中，您可预置把消息号分配给未来项目和库时所采用的方式(菜单命令**选项 > 自定义**)。在该标签中，您将确定是只把消息号分配给CPU，还是只分配给项目。如果您希望日后指定分配，那么，您也可选择“始终询问设置”。
如果在创建项目或库时，最初设置的缺省值“仅用于 CPU”或“仅用于项目”已激活，那么，您就不再能改变该项目或库的消息号分配的类型。

如果已经设置了“仅用于项目”消息号分配，希望设置“仅用于 CPU”分配，则可如下操作：

1. 在SIMATIC管理器中，选择相应的项目或库。
2. 选择菜单命令**文件 > 另存为**。
3. 在下一个对话框中启用“重新排列”复选框，并输入新的名称。
4. 使用“另存为”启动过程，并使用“确定”对条目进行确认。
5. 在随后出现的某个对话框中，可以指定“仅用于CPU”消息号分配。

您可使用**文件 > 删除**命令删除原来的项目或库。

16.2 组态项目消息

16.2.1 如何为项目分配消息号

由项目中唯一的编号来识别消息。为此，在总的可用范围(1 - 2097151)内，给每个STEP 7 程序分配一个编号范围。如果复制一个程序，并由此产生冲突 - 即如果已经在目标范围内分配相同的消息号 - 那么必须给新程序分配一个新的编号范围。如果出现该情况，那么 STEP 7 会自动打开一个可以指定新编号范围的对话框。

如果没有组态任何消息，那么也可以使用菜单命令**编辑 > 特殊对象属性 > 消息号**来设置或修改S7程序的编号范围。

缺省情况下，消息号范围以 20,000 的步长进行分配。

16.2.2 分配和编辑与块有关的消息

将与块有关的消息分配给块(背景数据块)。要创建一个与块有关的消息，可以使用系统功能块(SFB)和系统功能(SFC)作为消息块。

16.2.2.1 如何为项目创建与块相关的消息

基本步骤

为任务选择一个合适的消息块。

↓

将 FB 编程为消息类型块。

- 选择希望分配消息的功能块(FB)，然后打开该功能块。
- 填写变量声明表。
- 在 FB 的代码部分编写消息块调用程序。

↓

调用消息组态(标准对话框或 PCS7 消息组态)。

↓

在 FB 中创建具有文本、属性和显示设备的消息模板。

↓

将背景数据块与 FB 关联，然后根据要求进行修改。

↓

将所组态的数据传送到 WinCC 数据库。

编写消息类型块(FB)

1. 在SIMATIC管理器中，如果要为功能块生成一个与块有关的消息，那么双击打开该功能块(FB)。
   结果：在“LAD/STL/FBD”窗口中打开并显示选中的块。
2. 填写变量声明表。对于在功能块中调用的每个消息块，必须在调用功能块中声明变量。

   在变量概述栏中输入下列变量：

   - 在“IN”参数中写入一个符号名作为消息块的输入，例如，“ Meld01”(用于消息输入 01)，以及数据类型(必须为没有初始值的“DWORD”)。
   - 对于“STAT”参数，输入待调用消息块的符号名，例如，“alarm”和相应的数据类型，在本例中为“SFB33”。
3. 在功能块的代码部分，插入选中消息块的调用，在此为“CALL alarm”，然后用回车完成输入。

   结果：在功能块的代码部分显示所调用消息块(此处为SFB33)的输入变量。

4. 将在第2步中分配好的消息块的输入符号名(此处为“Mess01”)分配给变量“EV_ID”。现在，将系统属性应用于“alarm”类型的消息。

   结果：如果没有选择该栏，那么在参数“IN”的“名称”栏中应该出现一个标记。然后，选中的块被设置为消息类型块。并且自动分配所要求的系统属性(例如，S7_server和S7_a_type)以及相应的值(注意：对于特定的系统功能，需要手动分配参数“IN”的系统属性。为此，选择菜单命令**编辑 > 对象属性**，然后选择“属性”标签。)。

   警告：如果没有调用SFB，而调用包含多实例和已组态消息的FB，那么也必须在调用块中组态该FB (带多实例)的消息。

5. 对该功能块中消息块的所有调用重复步骤2 - 4。
6. 使用菜单命令**文件 > 保存**来保存块。
7. 关闭“LAD/STL/FBD”窗口。

打开消息组态对话框

- 选择期望的消息块，然后在SIMATIC管理器中选择菜单命令**编辑 > 特殊对象属性 > 消息**。

  结果：打开STEP 7消息组态对话框(标准对话框)。可以在PCS 7消息组态下获得关于打开PCS 7消息组态功能的信息。

编辑消息类型

1. 选择期望的消息块，打开消息组态，然后在“文本”和“属性”标签中输入所要求的消息文本或选择所要求的消息属性。
   如果选择一个多通道消息块(例如，“ALARM_8”)，那么，可以将特定文本分配给每个子编号，在一定程度上，也可以将特定属性分配给每个子编号。
2. 在“添加显示设备”对话框中单击“新设备”按钮，然后选择所需的显示设备，可以将所需的显示设备分配给消息模板。

在下列标签页中，输入显示设备所要求的文本和属性。点击“确定”，退出该对话框。

注意

在编辑显示设备特定的文本和属性时，请阅读随同显示设备提供的文档。

创建背景数据块

1. 已经创建了消息模板时，可以将背景数据块与该消息模板关联，然后编辑这些数据块的与背景有关的消息。
   为此，在SIMATIC管理器中双击、打开用于调用已组态功能块的块，例如“OB81”。在OB的开放式代码部分，输入调用(“CALL”)、待调用FB以及作为实例与FB关联的背景数据块的名称和编号。通过回车确认输入。

   实例：输入“CALL FB1, DB1”。如果不存在DB1，则对是否需要创建背景数据块的提示问题回答“是”。

   结果：创建背景数据块。在OB代码部分，显示相关FB的输入变量，本例中为“Mess01”，以及由系统分配的消息号，本例中为“1”。

2. 通过菜单命令**文件 > 保存**保存OB，然后关闭“LAD/STL/FBD”窗口。

编辑消息

1. 在SIMATIC管理器中，选择所生成的背景数据块，例如“DB1”，然后调用菜单命令**编辑 > 特殊对象属性 > 消息**来打开消息组态对话框。

   结果：打开“消息组态”对话框，然后显示选中的、带由系统分配的消息号的背景数据块。

2. 在相应的标签中输入相应背景数据块所要求的修改，并根据需要添加其他显示设备。点击“确定”，退出该对话框。

   结果：完成了所选背景数据块的消息组态。

传送组态数据

- 将已组态的数据传送到 WinCC 数据库(通过 AS-OS 连接组态)或 ProTool 数据库

16.2.2.2 如何为项目编辑与块相关的消息

1. 在SIMATIC管理器中，选择一个块，然后选择菜单命令**编辑 > 特殊对象属性> 消息**。
2. 在文件夹结构中，点击一个消息块输入或其中一个子编号(如果可用)。

   结果：显示标准消息的标签部分。
3. 在“文本”和“属性”标签中输入所要求的文本和属性。

   结果：已经创建了一个可以在所有显示设备上显示的标准消息。
4. 使用“新设备”按钮，添加一个“ProTool(Opx)”或“WinCC”类型的新显示设备。只能选择可显示已组态消息的那些显示设备。

   结果：添加并选择新设备，然后显示相应的标签部分。
5. 在特定显示的“文本”和“属性”标签中输入特定显示消息的属性和文本。

   结果：已经创建了一个仅在所选的显示设备上作为消息使用的消息变量。

如果希望编辑已存在的显示设备的其它消息变量：

- 在详细视图中通过双击选择并打开消息块。

  结果：自动选择第一个显示设备，现在可以为其编辑显示专用的消息变量。

16.2.2.3 如何为项目组态PCS 7消息

编辑将在 WinCC 显示设备上输出的消息模板和消息时，STEP 7 中的 PCS7 消息组态功能提供一种用户界面友好的方法：

- 简化显示设备组态(自动创建)
- 简化消息属性和文本输入
- 确保标准化消息。

打开**PCS7**消息组态功能

1. 在SIMATIC管理器中，选择希望编辑其消息文本的块(FB或DB)。选择菜单命令**编辑 > 对象属性**来打开用于输入系统属性的对话框。
2. 在所示的表中，输入系统属性“S7_alarm_ui”和值：“1”(值0禁止PCS7消息组态工具)。可在LAD/STL/FBD中设置属性参数。之后生成并分配给相应FB的DB采用这些属性，并且可独立于其消息类型(FB)进行切换。

注意

输入系统属性时，执行语法检查。以红色高亮显示错误的输入。

---

3. 点击“确定”，退出该对话框。
4. 选择菜单命令**编辑 > 特殊对象属性 > 消息**

   结果：打开“PCS7消息组态”对话框。

编辑消息类型

1. 在SIMATIC管理器中，选择希望编辑其消息文本的FB，然后打开PCS7消息组态对话框。

   结果：该对话框为每个在FB中有变量声明的消息块显示一个标签。
2. 填写消息组件“来源”、“OS区”和“批处理标识符”的文本框。
3. 给所使用消息块的所有事件输入消息等级和事件文本，然后指定是否必须单独应答每个事件。
4. 对于用于所有实例并禁止修改的消息部分，选择“锁定”复选框。

编辑消息

1. 打开SIMATIC管理器。选择希望编辑其消息文本的背景数据块，然后打开PCS7消息组态功能。
2. 不要编辑没有锁定的与实例有关的消息部分。

16.2.3 分配和编辑与符号相关的消息

16.2.3.1 如何为项目分配和编辑与符号相关的消息

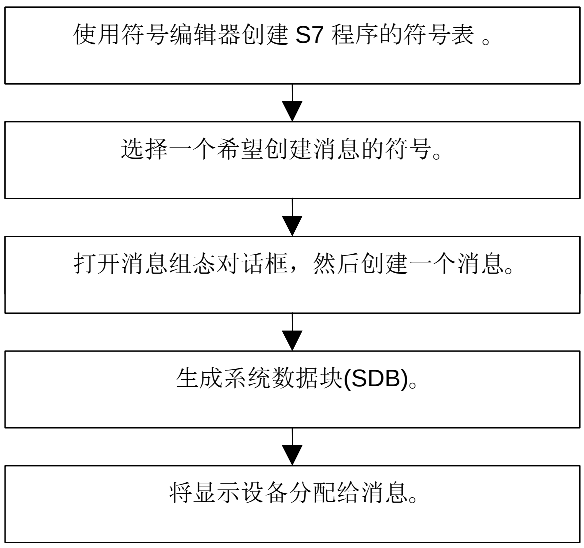
与符号有关的消息(SCAN)是直接分配给符号表中的信号的。信号都限制为布尔型地址：输入(I)、输出(Q)和位存储器(M)。可以通过消息组态功能给这些信号分配不同的属性、消息文本以及高达10个相关值。通过设置过滤器，可以更方便地在符号表中选择信号。

使用一个与符号有关的消息，可以在预设的时间间隔内扫描一个信号，确定该信号是否发生变化。

注意

时间间隔取决于所使用的CPU。

基本步骤



在处理过程中，已经组态消息的信号和程序的扫描是异步的。该信号以已设定的时间间隔进行扫描。在所分配的显示设备上显示消息。

警告

如果希望分配或编辑与符号有关的消息，且在同一个工作过程中，先前已经在两个符号表之间复制了符号，那么需要首先关闭不再操作的符号表。否则，将不能保存消息组态。在特定条件下，消息组态对话框中最后一个输入将丢失。

16.2.4 创建和编辑用户定义的诊断消息

使用此功能，可以在诊断缓冲区中写入用户条目，并发送在消息组态应用中创建的相应消息。用户自定义诊断消息通过系统功能 SFC52 (WR_USMSG; Error Class A or B)创建，该功能用作消息块。必须在用户程序中插入 SFC52 的调用，并为它分配事件 ID。

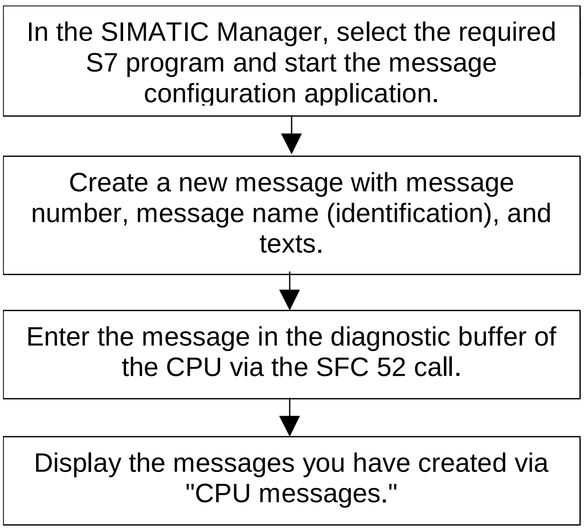
要求

在可以创建用户自定义诊断消息前，必须做好以下事项：

- 在 SIMATIC 管理器中创建项目
- 在项目中创建希望为其分配一个或更多消息的 S7/M7 程序。

基本步骤

要创建和显示用户自定义诊断消息，如下操作：



16.3 组态CPU消息

16.3.1 如何分配面向CPU的消息号

由一个唯一编号标识 CPU 消息。这通过给每个 CPU 分配一个编号区实现。除了将消息号分配给项目之外，没有必要将新编号区分配给新程序。因此不要求重新编译程序。请注意复制单个块的特例：此时，必须重新编译程序，以实现已修改的消息号。

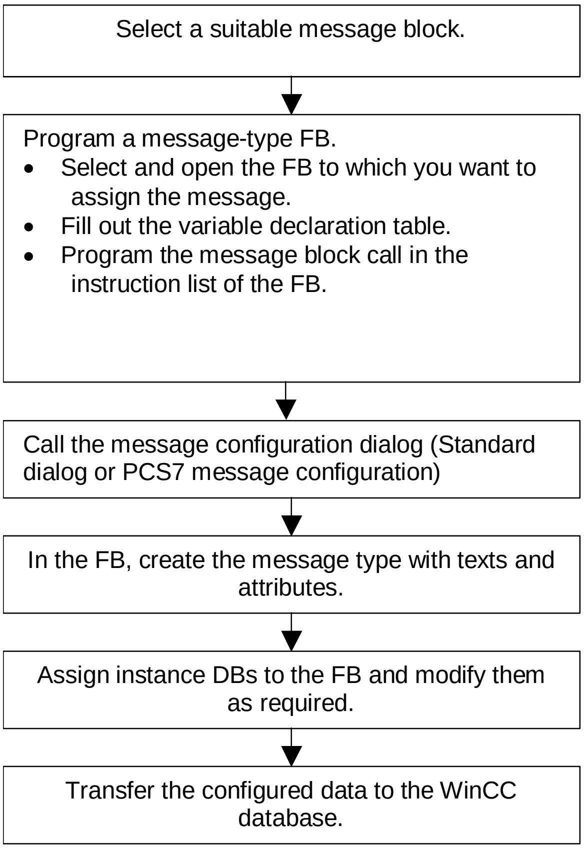
要求

- WinCC V6.0
- ProTool V6.0

16.3.2 分配和编辑与块有关的消息

16.3.2.1 如何为CPU创建与块相关的消息

操作流程



编写消息类型块(FB)

1. 在SIMATIC管理器中，选择要生成与块有关的消息的功能块(FB)，然后双击打开该块。
   结果：在“LAD/STL/FBD”窗口中打开并显示选中的块。
2. 填写变量声明表。对于功能块中调用的每个消息块，必须在调用的功能块中声明变量。

   在变量一览表中输入下列变量：

   - 在“IN”参数中，填入符号名作为消息块的输入，例如，“ Meld01”(用于消息输入 01)，以及数据类型(必须为没有初始值的“DWORD”)。

- 对于“STAT”参数，输入待调用消息块的符号名，例如，“alarm”和相应的数据类型，在该例中为“SFB33”。

3. 在功能块的代码部分，插入选中消息块的调用，在此为“CALL alarm”，然后用回车完成输入。

   结果：在功能块的代码部分显示所调用消息块(在此为SFB 33)的输入变量。

4. 将在第2步中分配好的消息块输入符号名(此处为“Mess01”)分配给变量“EV_ID”。

   结果：如果没有选择该栏，那么在参数“IN”的“名称”栏中应该出现一个标记。然后，选中的块被设置为消息类型块。并且自动分配所要求的系统属性(例如，S7_server和S7_a_type)以及相应的值(注意：对于特定的系统功能，需要手动分配参数“IN”的系统属性。为此，选择菜单命令**编辑 > 对象属性**，然后选择“属性”标签。)。

   警告：如果调用包含多背景和已组态消息的FB而不是SFB，那么也必须在调用块中组态该FB的消息。

5. 对该功能块中所有消息块的调用重复步骤2 - 4。
6. 使用菜单命令**文件 > 保存**保存块。
7. 关闭“LAD/STL/FBD”窗口。

打开消息组态对话框

- 在SIMATIC管理器中选择期望的消息块，然后选择菜单命令**编辑 > 特殊对象属性 > 消息**。

  结果：打开STEP 7消息组态对话框。可在PCS 7消息组态下查找打开PCS7消息组态功能的相关信息。

编辑消息类型

- 选择期望的消息块。
- 在相应的栏中输入所要求的文本或选择所要求的属性。
  在“消息组态”对话框中，点击“更多”按钮，然后在“缺省文本”标签中输入消息文本和附加文本。
  如果选择一个多通道消息块(例如，“ALARM_8”)，那么可以将特定文本分配给每个子编号，并在一定程度上，可以将特定属性分配给子编号。
- 如果背景的文本或属性禁止修改，那么请在消息类型中锁定它们。

创建背景数据块

1. 创建消息类型后，可以将背景数据块与该消息类型关联，然后可以给这些数据块编辑特定背景的消息。
   为此，在SIMATIC管理器中双击，打开调用已组态好的功能块的块，例如“OB81”。在OB的开放式代码部分，输入调用(“CALL”)、待调用的功能块的名称和编号，以及想要关联到这个功能块的背景数据块的名称和编号。通过回车确认输入。

   实例:输入“CALL FB1, DB1”。如果不存在DB1，则对是否需要创建背景数据块的提示确认为“是”。

   结果：创建背景数据块。在OB代码部分，显示关联FB的输入变量，本例中为“Mess01”，以及由系统分配的消息号，本例中为“1”。

2. 通过菜单命令**文件 > 保存**保存OB，然后关闭“LAD/STL/FBD”窗口。

编辑消息

1. 在SIMATIC管理器中，选择已生成的背景数据块，例如“DB1”，然后选择菜单命令**编辑 > 特殊对象属性 > 消息**来打开消息组态对话框。

   结果：打开“消息组态”对话框，然后显示选中的背景数据块，这些数据块包含系统分配的消息号。

2. 在相应的标签中输入相关联背景数据块所要求的修改，并根据需要添加其他显示设备。点击“确定”，退出该对话框。

   结果：完成了所选背景数据块的消息组态。

传送组态数据

- 将已组态的数据传送到 WinCC 数据库(通过 AS-OS 链接组态)或 ProTool 数据库

16.3.2.2 如何为CPU编辑与块相关的消息

1. 选择一个消息块，然后选择菜单命令**编辑 > 特殊对象属性 > 消息**来调用消息组态。
2. 在文件夹结构中，点击一个消息块输入或其中一个子编号(如果有的话)。
   还可以点击“更多”按钮，在“缺省文本”和“附加文本”对话框中输入所要求的文本(带换行符)。

   结果：已经创建了一个标准消息。

16.3.2.3 如何为CPU组态PCS 7消息

对于编辑在 WinCC 显示设备(从 V6.0 版本起)上输出的消息类型和消息，STEP 7 中的 PCS7 消息组态功能提供一种用户界面友好的方法：

- 简化的显示设备组态
- 简化的消息属性和文本输入
- 确保标准化消息。

打开PCS7消息组态功能

1. 在SIMATIC管理器中，选择希望编辑其消息文本的块(FB或DB)。选择菜单命令**编辑 > 对象属性**来打开用于输入系统属性的对话框。
2. 在所示的表中，输入系统属性“S7_alarm_ui”和值：“1”(值0表示禁止PCS7消息组态工具)。可在LAD/STL/FBD中设置属性参数。之后生成并分配给相应功能块的数据块自动接受这些设置，并可使用各自的属性设置做一些独立于消息类型(FB)的改变。

注意

输入系统属性时，将执行语法检查。以红色高亮表示输入错误。

---

3. 点击“确定”，退出该对话框。
4. 选择菜单命令**编辑 > 特殊对象属性 > 消息**

   结果：打开“PCS7消息组态”对话框。

编辑消息类型

1. 在SIMATIC管理器中，选择希望编辑其消息文本的FB，然后打开PCS7消息组态对话框。
2. 点击“更多”，打开“消息文本块”。填写消息组件“来源”、“OS区”和“批处理标识符”的文本框。
3. 给所用消息块的所有事件输入消息等级和事件文本，然后指定是否必须每个事件都必须应答。
4. 对于用于所有背景并禁止被修改的消息部分，选择“锁定”复选框。

编辑消息

1. 打开SIMATIC管理器。选择希望编辑其消息文本的背景数据块，然后打开PCS7消息组态功能。
2. 不要编辑没有锁定的与背景有关的消息部分。

16.3.3 分配和编辑与符号相关的消息

16.3.3.1 如何为CPU分配和编辑与符号相关的消息

与符号有关的消息(SCAN)直接分配给符号表中的一个信号。信号都限制为布尔型地址：输入(I)、输出(Q)和位存储器(M)。可以通过消息组态功能给这些信号分配不同的属性、消息文本以及高达10个相关值。通过设置过滤器，可以更方便地在符号表中选择信号。

使用一个与符号有关的消息，可以在预设的时间间隔内扫描一个信号，确定该信号是否发生变化。

注意

时间间隔取决于所使用的CPU。

基本步骤

使用符号编辑器创建 S7 程序的符号表。

↓

选择一个希望创建消息的符号。

↓

打开消息组态对话框，然后创建一个消息。

↓

生成系统数据块(SDB)。

↓

在处理过程中，已组态消息的信号的扫描是和程序异步的。是以预设的时间间隔进行扫描的。在所分配的显示设备上显示消息。

警告

如果希望分配或编辑与符号有关的消息，且在同一个工作过程中已经在两个符号表之间复制了符号，那么需要首先关闭不再需要工作的符号表。否则，不能保存消息组态。在特定条件下，将丢失消息组态对话框的最后一个输入。

16.3.4 创建和编辑用户定义的诊断消息

使用此功能，可以在诊断缓冲区中写入用户条目，并发送在消息组态应用中创建的相应消息。用户自定义诊断消息通过系统功能 SFC52 (WR_USMSG; Error Class A or B)创建，该功能用作消息块。必须在用户程序中插入 SFC52 的调用，并为它分配事件 ID。

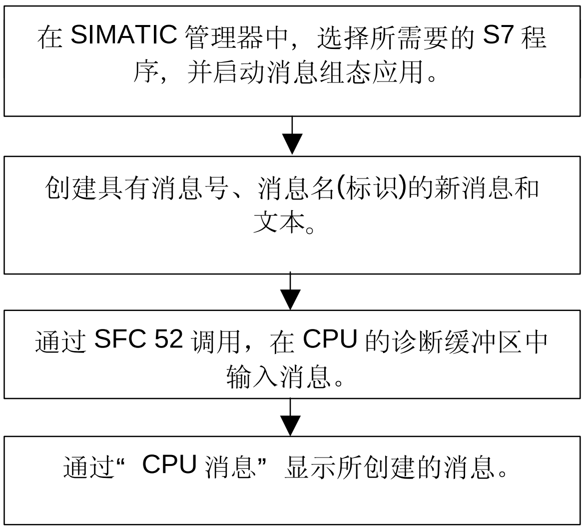
要求

在可以创建用户自定义诊断消息前，必须做好以下事项：

- 在 SIMATIC 管理器中创建项目
- 在项目中创建希望为其分配一个或更多消息的 S7/M7 程序。

基本步骤

要创建和显示用户自定义诊断消息，如下操作：



16.4 编辑消息时的提示

16.4.1 将相关值添加到消息中

要将当前信息(如来自某个过程)添加到块相关的和符号相关的消息中，可以在消息文本中的任意点插入相关的值。

要添加值，如下进行操作：

1. 创建具有下列结构的块：
   @<相关值的编号><元素类型><格式代码>@。
2. 将这个块插入到消息文本中要显示相关值的位置上。

元素类型

此参数为相关值数据类型分配唯一的标识：

元素类型	数据类型
Y	BYTE
W	WORD
X	DWORD
I	Integer
D	Integer
B	BOOL
C	CHAR
R	REAL

元素类型仅指定由 PLC 传送的数据类型。它不作为计算算子使用。

格式代码

这些代码指定相关值在显示设备上的输出格式。格式指令由“%”符号引入。对于消息文本，有下列固定的消息代码：

格式代码	描述
%[i]X	具有i索引的十六进制值
%[i]u	具有i索引的无符号 十进制值
%[i]d	具有i索引的有符号 十进制值
%[i]b	具有i索引的二进制值

格式代码	描述
%[i][.y]f	整数(固定点编号) 有符号值，格式为 [-]dddd.dddd dddd：一个或多个数字，小数点后y位，共i位
%[i]s	i位字符串(ANSI字符串) 字符被打印到第一个0字节(十六进制00)。
%t#<文本库的名称>	访问文本库。

如果格式代码太小，值仍旧以其全部长度输出。

如果格式代码太大，在值前输出合适的空格数目。

注意

注意，也可以选择指定“[i]”，在这种情况下，当输入此参数时必须省略括号。

相关值的实例

@1I%6d@：来自相关值 1 的值显示为十进制数字，最多具有 6 位。

@2R%6f@：例如，来自相关值 2 的值“5.4”，显示为一个整数“5.4”(三个前导括号)。

@2R%2f@：例如，来自相关值 2 的值“5.4”，显示为整数“5.4”(位数太小时不切断)。

@1W%t#Textbib1@：数据类型 WORD 的相关值 1 是索引，要使用的文本通过它在 TextLib1 文本库中被参考。

注意

当使用S7-PDIAG时，必须始终用“X”指代元素类型。
如果希望向一个ALARM_S块传送一个以上的相关值，可以发送一个最大长度为12字节的数组。例如，可以是最多12个字节或字符，最多6个字或整数，或最多3个双字、实数或双整数。

16.4.2 将文本库中的文本集成到消息中

可以从最多四个不同的文本库中随意将文本集成到一个消息中。文本可以自由放置，以保证能在外语消息中使用。

操作过程如下：

1. 在SIMATIC管理器中，选择CPU或CPU从属对象，并选择菜单命令**选项 > 文本库 > 系统文本库**或**选项 > 文本库 > 用户指定的文本库**，打开文本库。

警告

如果已选择将消息号分配给CPU，可以仅将用户文本库中的文本集成到消息中。

---

2. 确定希望集成的文本索引。
3. 在消息中希望显示文本的位置处，输入格式为@[Index]%t#[Textbib]@的占位符。

注意

[索引] = 1W，此处1W是用于WORD类型消息的第一个关联值。

实例

组态的消息文本：压力上升@2W%t#Textbib1@

名称为 Textbib1 的文本库：

索引	德语	英语
1734	zu hoch	too high

为所传送的第二个关联值分配了数值 1734。显示下列消息：压力上升太高。

16.4.3 删除相关值

通过在表示相关值的消息文本中删除字符串，删除相关值。

操作过程如下：

1. 在消息文本中定位对应希望删除相关值的信息块。
   该块以@符号开始，其后为识别相关值的位置指示符以及格式代码；以另一个@符号结束。
2. 从消息文本中删除该信息。

16.5 翻译和编辑与操作员有关的文本

过程编辑期间在显示设备上输出的文本，通常将以自动化解决方案编程时使用的语言进行输入。

经常可能出现的情况是，将对显示设备上的消息进行响应的操作员不会使用这种语言。该用户需要以其母语书写的文本才能确保顺利无误地进行处理并对系统输出的消息进行快速响应。

STEP 7允许您将某些文本以及所有与操作员相关的文本翻译为所需要的语言。其唯一的先决条件就是您已在项目中安装了这种语言(菜单命令：SIMATIC管理器中的**选项 > 用于设备显示的语言**)。可用语言的数目在安装Windows时确定(系统属性)。

采用该方式，可确保将来面对这种消息的任何用户都可使这些消息以适当的语言进行显示。该系统特性将极大地增加过程处理的安全性和精确性。

与操作员相关的文本就是用户文本和文本库。

16.5.1 转换和编辑用户文本

您可以为整个项目、为 S7 程序、块文件夹或单个块，以及为符号表对象创建用户文本，条件是在这些对象中组态了消息。例如，它们包含可以在显示设备中显示的所有文本和消息。对于一个项目，可以有多个可以翻译成所需语言的操作者相关文本的列表。

可以选择项目中已有的语言(菜单命令**选项 > 显示设备的语言…**)。以后可添加或删除语言。

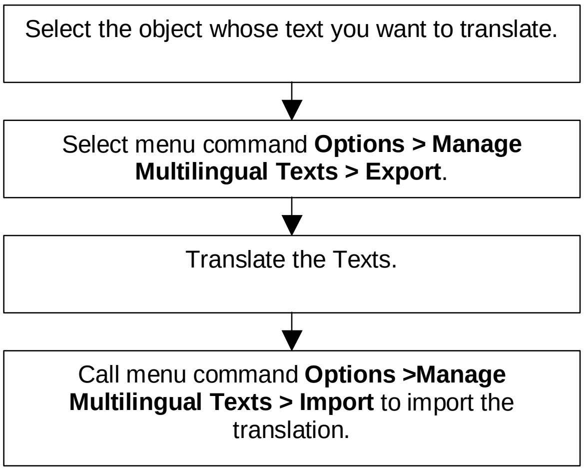
导出和导入操作者相关的文本

可以在STEP 7外翻译或编辑在STEP 7中创建的操作者相关文本。为此，将所显示的操作者相关文本导出到导出文件中，用基于ASCII的编辑器或电子表格工具，如Microsoft EXCEL编辑此导出文件(菜单命令**选项 > 管理多语言文本 > 导出**)。打开文件后，画面显示包含各种语言列的表格。第一列始终显示所设置的标准语言。翻译好文本后，再次将它们重新导入到STEP 7中。

只能将操作者相关文本导入到项目中导出文本的部分。

基本步骤

确保已经在SIMATIC管理器中，按照菜单命令**选项 > 显示设备的语言**设置了文本翻译目标语言。



注意

只能在用于翻译的应用程序下打印用户文本。

16.6 翻译和编辑文本库

16.6.1 用户文本库

用户文本库允许根据相关值动态查看文本或文本段。在此，相关值提供当前文本的文本库索引。在该位置输入的占位符为待显示的动态文本。

可以为程序创建用户库，并在其中输入文本和选择个人索引。该应用程序将自动检查用户库中索引的唯一性。该 CPU 可用的所有消息都可以包含对用户文本库的交叉引用。

文本库文件夹中文本库的数目不受限制。因此，举个例子，可以将同一个程序用于不同的控制任务，并仅根据应用场合来使用文本库。

警告

将一个包含对文本库交叉引用的消息类型块复制到另一个程序中时，必须带上相应的文本库，或创建具有相同名称的新文本库，或在消息文本中编辑交叉引用。

创建文本项时，缺省情况下始终分配一个索引。输入一个新行时，应用程序将下一个空闲索引作为缺省值。文本库中禁止出现模糊索引，应用程序拒绝该类索引。

16.6.2 系统文本库

系统文本库是在创建块时自动生成的，例如，在“系统错误 报告”中。用户不能创建系统文本库，只能编辑已存在的文本库。

该 CPU 可用的所有消息都可以包含对文本库的交叉引用。

16.6.3 翻译文本库

系统文本库和用户文本库提供文本的列表，这些文本可以集成到消息中、在运行期间动态更新，以及在编程设备或其它显示设备上显示。

系统文本库中的文本由 STEP 7 或 STEP 7 选项包提供。可以将多个文本库分配给一个 CPU。可以将这些文本翻译成所需要的语言。

在 SIMATIC 管理器中，可以选择项目中已有的语言(菜单命令**选项 > 显示设备的语言…**)。以后可添加或删除语言。

当开始翻译文本库(菜单命令 **选项 > 管理多语言文本 > 导出**)时，将生成可编辑的导出文件，例如可以在 Microsoft EXCEL 下编辑。打开文件后，画面显示包含各种语言列的表格。

警告

决不要通过双击打开*.cvs导出文件。始终在Microsoft EXCEL中使用菜单命令**文件 > 打开**来打开该文件。

注意

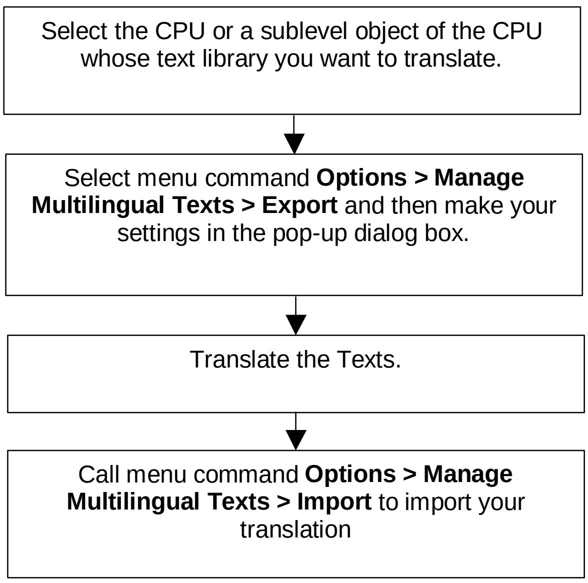
只能在用于翻译的应用程序中打印用户文本。

导出文件的实例

德语	英语
ausgefallen	Failure
gestört	Disruption
Parametrierfehler	Faulty parameter assignment

基本步骤

在SIMATIC管理器中，使用菜单命令**选项 > 显示设备的语言…**，确保已设置了希望的文本库翻译语言。



16.7 将消息组态据传送到可编程控制器

16.7.1 将组态数据传送到可编程控制器

概述

使用传送程序 AS-OS Engineering 将所生成的消息组态数据传送到 WinCC 数据库。

要求

启动传送之前，必须满足下列要求：

- 已经安装“ASOS Engineering”。
- 已经生成用于创建消息的组态数据。

基本步骤

在 SIMATIC 管理器中，创建希望显示消息的 OS 对象。

↓

使用菜单命令选项 **> AS-OS** 连接数据 **>** 传送，启动组态数据的传送。

16.8 显示CPU消息和用户自定义的诊断消息

使用“CPU 消息”功能(菜单命令 **PLC > CPU** 消息)，您可显示关于诊断事件的非同步消息和自定义的诊断消息以及来自 ALARM_S 块(SFC 18 和 SFC 108，用于生成与块相关的始终进行应答的消息；SFC 17 和 SFC 107，用于生成与块相关的有应答的消息)的消息。

也可使用菜单命令**编辑 > 消息 > 自定义诊断**来启动 CPU 消息应用程序中的消息组态应用程序，并创建自定义的诊断消息。其条件就是通过在线项目启动了 CPU 消息应用程序。

显示选项

使用“CPU 消息”功能，您可决定是否以及如何显示所选 CPU 的在线消息。

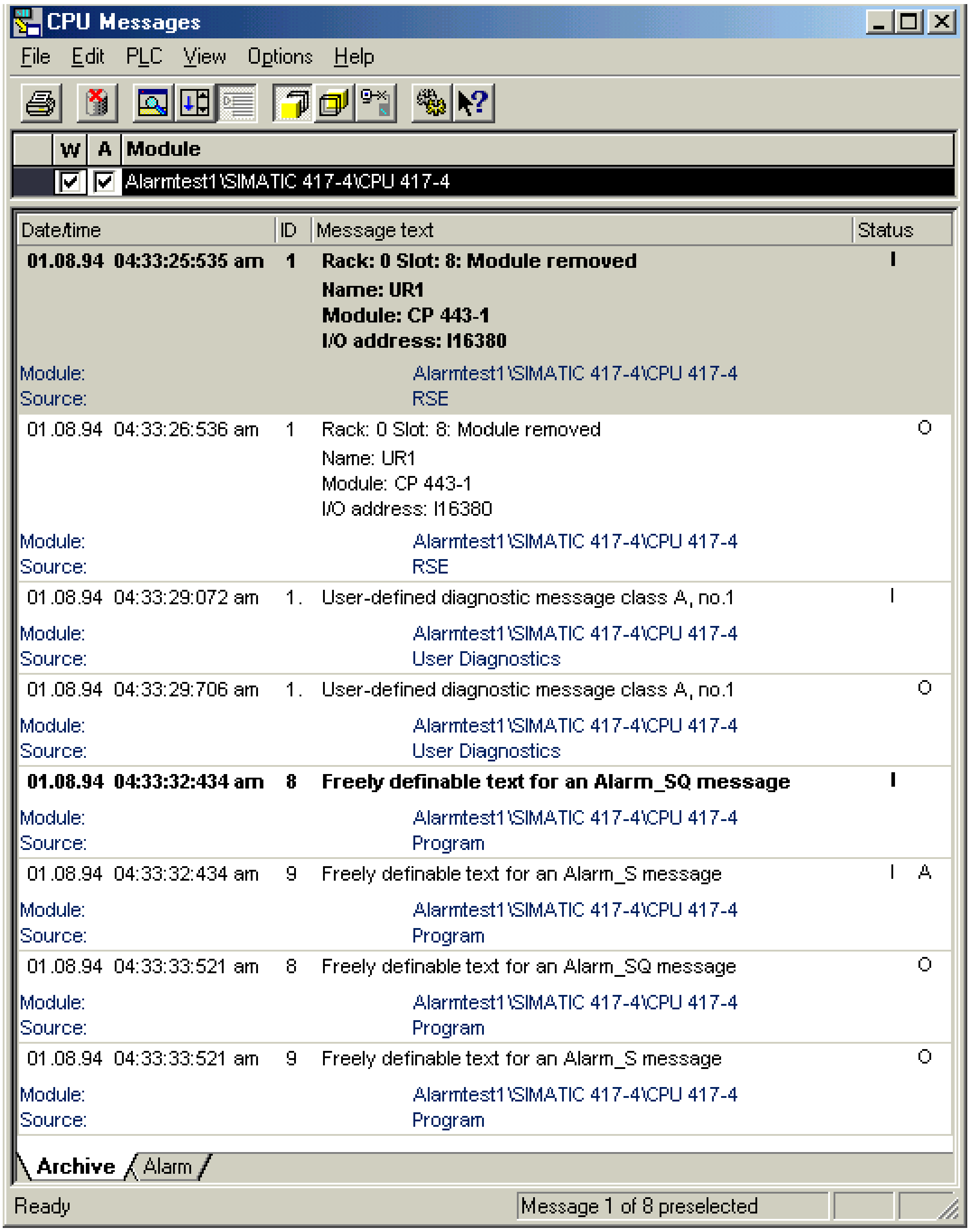
- **“处于前台”**：包含有 CPU 消息的窗口将出现在前台。每当接收一条新的消息后，窗口都将出现在前台。
- **“留在后台”**：在后台接收 CPU 消息。在接收新的消息时，窗口将仍然位于后台，如果需要，也可将其置于前台。
- **“忽略消息”**：新的 CPU 消息将**不**显示，而且，与其它两个模式相反，将**不**进行归档。

在“CPU消息”窗口中，您可选择“归档”标签或“中断”标签。在两个标签中，您都可选择菜单命令**视图 > 显示信息文本**以指定显示消息时是否带有或不带有信息文本。用户可根据需要对栏目进行整理。

“归档”标签

进入的消息将在这里进行显示和归档，并根据事件消息时间进行排序。归档的容量(40 和 3000 之间的 CPU 消息)可通过“设置 - CPU 消息”对话框中的菜单命令**选项 > 设置**进行设置。如果超出了所设置的归档容量，将删除队列中最旧的消息。

可确认的消息(ALARM_SQ 和 ALARM_DQ)均将以黑体字显示。您可在菜单命令**编辑 > 确认 CPU 消息**下确认这些消息。



“中断”标签

ALARM_S 块中尚未接收或确认的排队消息的状态也将显示在“中断”标签中。

您可选择菜单命令**视图 > 多行消息**来显示一行或多行的消息。此外，您可根据需要对栏目进行排序。

更新 ALARM_S 块中的消息

在更新期间，所有未发送或未确认的消息均将再次输入到归档中。在下列情况下，将对消息进行更新：

- 如果在与消息相关的模块上执行重新启动(非冷启动)
- 如果在模块列表中点击 ALARM_S 块消息的选项“A”。

基本步骤

为所选模块组态 CPU 消息：

在 SIMATIC 管理器中，选择 S7 程序，并选择菜单命令 **PLC > CPU** 消息。

↓

指定您希望接收的消息以及如何显示这些消息。

16.8.1 组态CPU消息

要组态所选模块的 CPU 消息，可如下进行操作：

1. 在SIMATIC管理器中，通过在线项目启动CPU消息应用程序。为此，选择一个在线的S7程序，然后使用菜单命令PLC > CPU**消息**调用所选CPU的CPU消息应用程序。

   结果：将出现“CPU消息”应用程序窗口，列出已注册CPU的。
2. 通过重复步骤1可添加用于其它程序或接口的已注册CPU来扩展列表。
3. 点击列表条目前面的复选框，指定模块要收的消息：

   A：激活ALARM_S块(用于生成始终进行确认的块相关信息的SFC 18和SFC 108以及用于生成可确认的块相关信息的SFC 17和SFC 107)中的消息，例如，S7 PDIAG、S7-GRAPH中的报告过程诊断信息，或系统错误。

   W：激活诊断事件。
4. 设置文档的大小。

   结果：只要上述消息一发生，就将它们写入消息文档，并按照您选择的方式进行显示。

注意

在SIMATIC管理器中为其调用了菜单命令**PLC > CPU消息**的CPU均将输入到“CPU消息”应用程序窗口中的已注册模块列表中。列表中的条目将保留，直到在“CPU消息”应用程序窗口中将其删除。

16.8.2 显示所存储的CPU消息

除非选择了菜单命令**视图 > 忽略消息**，归档始终记录CPU消息。始终显示所有已归档的消息。

16.9 组态“系统错误报告”

引言

当发生系统错误时，S7 组件和 DP 标准从站(属性由其 GSD 文件确定的从站)可以触发组织块调用。

实例：如果有断线，具有诊断能力的模块可以触发诊断中断(OB82)。

对于发生的系统错误，S7 组件提供信息。启动事件信息，即所分配的 OB 的本地数据(包含数据记录 0 及其它)，提供关于错误位置(如模块的逻辑地址)和类型(如通道错误或备份故障)的常规信息。

此外，可以通过附加诊断信息(用 SFC51 读数据记录 1 或用 SFC13 读 DP 标准从站的诊断消息)更详细地说明错误。这种情况的实例是通道 0 或 1 和接线断开或测量范围超出限度。

有了报告系统错误功能，STEP 7 就可提供方便的方法，以消息形式显示组件提供的诊断信息。

必需的块和消息文本由 STEP 7 自动生成。用户要做的就是将所生成的块装载到 CPU 中，并将文本传送到所连接的 HMI 设备上。

在所支持的组件和功能范围一节中可找到所支持的各种从站诊断信息的总览。

基本步骤

组态站并将参数分配给模块(启用诊断中断)。

↓

设置报告系统故障并组态消息。

↓

开始生成块。

↓

检查和补充所生成的块。

↓

将所生成的消息的组态数据传送到合适的 HMI 设备，该设备可产生系统故障报告。

↓

将所生成的块下载到 CPU。

消息通过 ALARM_S/SQ 标准消息路径发送到编程设备上的 CPU 消息中，或发送到所连接的 HMI 设备上。

16.9.1 所支持的组件和功能范围

报告系统错误(功能)支持 S7 300 站、S7 400 站、DP 从站和 WinAC 组件，条件是它们支持诊断中断插入/删除模块中断，以及特定通道诊断之类的功能。

报告系统错误(功能)不支持下列组件：

- S7-300 站中的 DP 主站接口模块(CP 342-5 DP)上的 M7、C7 和 PROFIBUS-DP 组态。

在重启动的情况下，还必须注意可能发生丢失中断消息。这是因为在重启动期间，CPU 的消息确认存储器不能被删除，但报告系统错误(功能)会复位内部数据。

在下列的两个表中，将找到“报告系统错误”所支持的各种从站的诊断块。

诊断块	标识号 (故障插槽)	通道标志 (通道错误) 1)	模块状态 (模块错误，不正确/无模块)	设备标志
标题ID 2)	**0x01**	**0x10**	**0x00 类型0x82**	**0x00 + 1字节 诊断信息**
ET 200S	消息： “诊断可用”	纯文本消息	纯文本消息	-
ET 200M	未估计	未估计	未估计	-
ET 200X	消息： “诊断可用”	-	-	-
ET 200X DESINA	消息： “诊断可用”	纯文本消息	纯文本消息	-
ET 200L	未估计	-		-
ET 200B Digital				消息： “模块故障”
ET 200B Analog		-	-	
ET 200C Digital				
ET 200 C Analog	消息： “诊断可用”			消息： “模块故障”
ET 200 U	消息： “诊断可用”			消息： “模块故障”
ET 200 iS	消息： “诊断可用”	纯文本消息	纯文本消息	
DP AS-i Link	消息： “诊断数据可用”	-	纯文本消息	-

1) DS0：标准诊断，例如模块故障、外部辅助电压或前连接器丢失， 长度为4字节，包含在OB12的本地数据中。
DS1：通道错误，随通道类型有不同定义，在用户程序中通过SFC 51可读。
文本来自S7 HW诊断。

2) 标题标识符：诊断消息中的标识符，标识不同的诊断部分。

诊断块	DS0/DS1 1)	其它实例
标题ID 2)	**0x00 类型0x01**	**0x00 其它类型**
ET 200S		
ET 200M	纯文本消息	未估计
ET 200X		
ET 200X DESINA	纯文本消息	
ET 200L	纯文本消息	
ET 200B Digital		
ET 200B Analog	纯文本消息	

诊断块	**DS0/DS1 [1]**	其它实例
ET 200 C Digital		
ET 200 C Analog	纯文本消息	
ET 200 iS	纯文本消息	
DP AS-i Link	消息：“模块错误”	

[1] DS0：标准诊断，例如模块故障、外部辅助电压或前连接器丢失，长度为4字节，包含在OB12的本地数据中。
DS1：通道错误，随通道类型有不同定义，在用户程序中通过SFC 51可读。
文本来自S7 HW诊断。

[2] 标题标识符：诊断消息中的标识符，标识不同的诊断部分。

诊断消息(也称标准从站消息)由上述诊断块组成，可在用户程序中通过 SFC 13 读取。

在 STEP 7 中，在“HW Config”(诊断硬件)在线窗口的“DP 从站诊断”标签卡的“十六进制显示”下调用模块状态，可显示诊断消息。

诊断中继器：诊断中继器的消息作为纯文本输出。文本从 GSD 文件中读取。

16.9.2 “报告系统错误”设置

有几种调用设置对话框的可能方法：

- 在 HW Config 中，选择希望为其组态系统错误报告的 CPU。然后选择菜单命令**选项 > 报告系统错误**。
- 如果已经生成用于报告系统错误的块，双击所生成的块(FB、DB)可调用对话框。
- 在站的属性对话框中，选择选项，在保存和编译组态期间自动调用。

如下到达保存和编译期间自动调用选项：

1. 在SIMATIC管理器中，选择合适的站。
2. 选择菜单命令**编辑 > 对象属性**。
3. 选择“设置”标签。

注意

也可以通过菜单命令**站 > 属性**，打开HW Config中的属性对话框的“设置”标签。

在对话框中，除了其它内容，请输入下列内容：

- 应生成哪个 FB 和哪个已分配的背景数据块
- 是否应生成参考数据
- 在报告系统错误生成期间是否始终显示警告
- 在保存和编译组态(参见以上的设置)后、自动调用报告系统错误时，是否应显示对话框。
- 生成错误 OB：尚不可使用的错误 OB 是否应在 S7 程序中生成，以及在哪个 OB 中调用“报告系统错误”。
- 出错时 CPU 的特性：可以确定哪个错误级别事件会触发 CPU 到 STOP 模式。
- 消息的显示(可能的文本部分的结构和次序)
- 消息是否应是可确认的
- 用户块接口应包含哪些参数

在所调用对话框的帮助中可以找到更详细的信息。

16.9.3 生成用于报告系统错误的块

在完成用于报告系统错误的设置后，可以生成所需要的块(FB 和 DB，根据组态可包括尚不存在的 DB)。为此，在“报告系统错误”对话框中，点击“生成”按钮。

生成下列块：

- 诊断FB (缺省：FB49)
- 诊断FB的背景数据块(缺省：DB49)
- 错误OB (如果已在“OB组态”对话框中选择了本选项)，
- 由诊断FB调用的可选用户块

16.9.4 所生成的错误OB

可以生成下列可“报告系统错误”的错误 OB：

- OB81 (电源错误)，可调用所生成的诊断FB。
- OB82 (诊断中断OB)，可调用所生成的诊断FB。
- OB83 (插入/删除中断)，可调用所生成的诊断FB。
- OB84 (CPU硬件故障)
  此OB生成时没有内容，所以当发生通讯错误时，CPU不切换到STOP模式(例如，插入和拆除MPI电缆时的MPI端接电阻器问题)。不估计错误，不生成消息。

- OB85 (程序执行错误)
  只有在出现更新过程映像的错误时(例如，移动模块)，才会禁止CPU切换到STOP。这样才能处理OB83中的诊断FB。任何报告系统错误消息后的CPU STOP设置均在OB83中生效。对于所有其它的OB85错误事件，CPU进入STOP模式。
- OB86 (扩展机架、DP主站系统或分布式I/O设备的故障)，可调用所生成的诊断FB。

如果错误 OB 已存在...

存在的错误OB不被重写。如果需要，会附加调用诊断FB。

如果组态包括分布式 I/O 设备...

对于在分布式I/O内的估计错误，所生成的FB会自动调用SFC13 (读DP从站的诊断数据)。为保证此功能，所生成的FB必须要么仅在OB1中调用，要么在具有短时间周期的循环中断OB和启动OB中调用。

注意

请注意下列各项：

- 当报告系统错误(功能)因“更新过程映像时出错”错误事件而生成OB85时，CPU不再进入STOP模式。
- 当下列错误产生时，CPU也会调用OB85：
  - 未装载的 OB 的错误事件
  - 当调用或访问未装载的 OB 时的错误

  当这些错误产生时，当报告系统错误(功能)生成OB85时，CPU仍进入STOP模式，就像使用“报告系统错误”之前一样。
- 对于OB84和OB85来说，“执行诊断FB后CPU进入STOP模式”设置不起作用，因为这些OB不调用报告系统错误的FB。在使用OB85的情况下，该设置由OB83中的FB调用间接提醒注意。

16.9.5 所生成的FB、DB

所生成的 FB 估计错误 OB 的本地数据，读取触发故障的 S7 组件的任何附加诊断信息，并自动生成合适的消息。

FB具有下列属性：

- 生成RSE (报告系统错误)的语言(也适用于所生成的背景数据块)

- 受保护的知识产权(也适用于所生成的背景数据块)
- 在运行时延迟到达中断
- 通过双击调用用于设置“报告系统错误”功能的对话框(也适用于所生成的背景数据块)。

用户块

因为诊断FB的知识产权受到保护，所以不能编辑它。不过，FB提供用户程序接口，所以可以访问诸如错误状态或消息号之类的信息。
用所选择的参数在所生成的FB中调用用于在用户程序(可以在对话框的用户块标签中设置)中进行估计的块。可使用下列参数：

名称	数据类型	注释
EV_C	BOOL	//消息到来(TRUE)或离去(FALSE)
EV_ID	DWORD	//所生成的消息号
IO_Flag	BYTE	//输入模块：B#16#54输出模块：B#16#55
logAdr	WORD	//逻辑地址
TextlistId	WORD	//文本库的ID (缺省文本库=1)
ErrorNo	WORD	//所生成的错误编号
Channel_Error	BOOL	//通道错误(TRUE)
ChannelNo	WORD	//通道编号
ErrClass	WORD	//错误类别
HErrClass	WORD	//H系统的错误级别

如果用户FB尚不存在，则由SFM用所选择的参数创建。

针对标准错误生成的错误文本按如下方式排列：

错误号		受影响的错误OB	在OB中的错误代码	
从	到		从	到
1	86	OB 72	B#16#1	B#16#56
162	163	OB 70	B#16#A2	B#16#A3
193	194	OB 72	B#16#C1	B#16#C2
224		OB 73	B#16#E0	
289	307	OB 81	B#16#21	B#16#33
513	540	OB 82		
865	900	OB 83	B#16#61	B#16#84
1729	1763	OB 86	B#16#C1	B#16#C8

大于12288的错误号称为通道错误。如果以十六进制形式查看错误号，可以计算通道类型并识别错误位。有关确切的描述，请参阅相应的模块帮助或通道帮助文本。

实例：

12288 = W#16#3000 -> 高位字节 0x30 - 0x10 = 通道类型 0x20 (CP 接口)；
低位字节 0x00，表示错误位 0

32774 = W#16#8006 -> 高位字节 0x80 - 0x10 = 通道类型 0x70 (数字输入)；
低位字节 0x06，表示错误位 6

17 控制和监视变量

17.1 组态用于操作员监控的变量

概述

STEP 7 提供了一种用户界面友好的对使用 WinCC 的过程或可编程控制器中的变量进行监控的方法。

该方法优于先前方法的地方是您将不再需要为每个操作员站(OS)单独组态数据，您只需使用STEP 7组态一次即可。在您使用STEP 7进行组态时，您可使用传送程序AS-OS EngineerinG (软件包“过程控制系统PCS 7”的一部分)将生成的数据传送到WinCC数据库，在这期间，将对数据的一致性及其与显示系统的兼容性进行检查。WinCC将使用变量块和图形对象中的数据。
您可使用 STEP 7 组态或修改下列变量的操作员监控属性：

- 输入、输出、以及功能块的入口/出口参数
- 位存储器和I/O信号
- CFC图中CFC块的参数

基本步骤

操作员监控变量的组态步骤取决于选择的编程/组态语言以及您希望监视和控制的变量的类型。然而，基本过程将始终包括有下列步骤：

1. 将操作员监控的系统属性分配给功能块的参数或符号表中的符号。

   在CFC中将不需要该步骤，因为您采用了从库中已经准备完毕的块。

2. 在对话框(S7_m_c)中为您希望控制和监视的变量分配所需的属性和记录属性。在操作员接口对话框(菜单命令**编辑 > 特定对象属性 > 操作员接口**)中，您可改变WinCC属性，例如限制值、替代值、以及协议属性等等。

3. 通过AS-OS Engineering工具将STEP 7生成的组态数据传送到显示系统(WinCC)。

命名惯例

对于 WinCC 要保存和传送的组态数据，它们将以 STEP 7 自动分配的唯一名称进行存储。操作员监控变量、CFC 图、以及 S7 程序等的名称均构成了该名称的一部分，因此也要服从某些惯例：

- 在S7项目中，S7程序的名称必须是唯一的(不同的站不能包含有具有相同名称的S7程序)。
- 变量、S7程序、以及CFC图的名称都不能包含有下划线、空格、或下列特殊字符：
  [‘] [.] [%] [-] [/] [*] [+]。

17.2 利用语句表、梯形图和功能块图表组态操作员监控属性

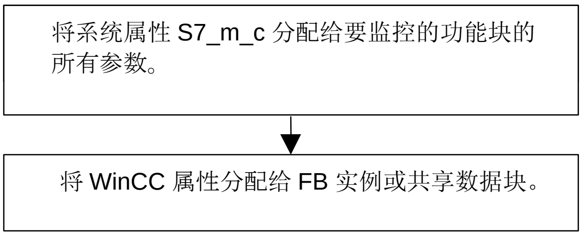
概述

按照下述过程，您可以使功能块参数能适用于操作员监控，并将所需要的O、C和M属性分配给用户程序中的相关背景数据块或者共享数据块。

要求

您必须已经创建了一个STEP 7项目、一个S7程序和一个功能块。

基本步骤



17.3 通过符号表组态操作员监控属性

概述

不管使用哪种编程语言，您都可以采用下述步骤来组态以下变量：

- 位存储器
- I/O信号

要求

开始组态之前，必须具备下列条件：

- 您已经在SIMATIC管理器中创建了一个项目。
- 该项目中必须有一个带符号表的S7程序。
- 必须打开符号表。

基本步骤

将“操作员监控”特殊对象属性分配给符号。

↓

将 WinCC 属性分配给符号。

17.4 使用CFC改变操作员监控属性

概述

使用 CFC 时，您可以从库里选择已经具有操作员监控属性的块，将它们放到图中并连接，从而创建自己的用户程序。

要求

您已经在 STEP 7 项目中插入了一个 S7 程序，创建了 CFC 图表，并将块放到该图中。

基本步骤

编辑块的对象属性。

注意

如果使用的是自己创建的块，并且已经给这些块分配了系统属性S7_m_c，只要再激活对话框“操作员监控”(菜单命令：**编辑 > 特殊对象属性 > 操作员监控**)中的“操作员监控”复选框，就能使这些块具有操作员监控属性。

17.5 将可编程控制器的组态数据传送给操作员接口

介绍

使用AS-OS Engineering传送程序，将所生成的操作员监控组态数据传送到WinCC数据库。

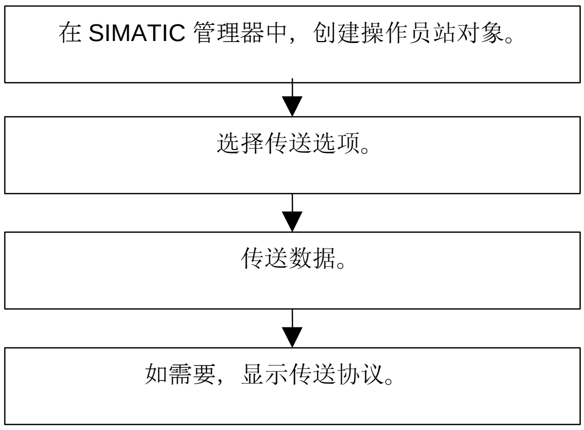
要求

开始传送之前，必须满足下列要求：

- 您已经安装了ASOS Engineering程序。
- 您已经生成了操作员监控组态数据。

基本步骤

要将操作员监控组态数据传送到WinCC数据库，可如下操作：



18 建立在线连接并标记CPU设置

18.1 建立在线连接

需要在编程设备和可编程逻辑控制器之间建立一个在线连接，用于下载 S7 用户程序/块、从 S7 可编程控制器中将块上传到编程设备，以及用于其它活动：

- 调试用户程序
- 显示和改变CPU的工作模式
- 显示并设置CPU的时间与日期
- 显示模块信息
- 在线和离线比较块
- 诊断硬件

为了建立在线连接，必须通过合适的接口连接编程设备和可编程逻辑控制器(例如，多点接口(MPI))。然后可通过项目的在线窗口或“可访问节点”窗口访问可编程控制器。

18.1.1 通过“可访问节点”窗口建立在线连接

这类访问能使您快速访问可编程逻辑控制器，以用于诸如测试目的。可以访问网络中所有可访问的可编程模块。如果在编程设备上没有关于可编程控制器的项目数据，请选择此方式。

使用菜单命令 **PLC > 显示可访问节点**，打开“可访问节点”窗口。在“可访问节点”窗口中，将显示网络中所有可访问的节点及其地址。

在“可访问节点”窗口中，还能显示不能用STEP 7 (例如编程设备或操作面板)编程的节点。

在括号中还可显示下列附加信息：

- (直接)：该节点直接连接到编程设备(编程设备或PC)。
- (无源)：不能通过PROFIBUS DP对该节点进行编程和状态修改。
- (等待)：不能与该节点进行通讯，因为其组态与网络中其它设置不匹配。

找到直接连接的节点

附加信息“直接”不支持 PROFInet 节点。为了仍旧能够找到直接连接的节点，请选择 PLC > 诊断/设置 > 节点闪烁测试菜单命令。

在显示的对话框中，可以设置闪烁持续时间，然后启动闪烁测试。直接连接的节点将由一个闪烁强制 LED 进行识别。

如果强制功能激活，则不能进行闪烁测试。

18.1.2 通过项目的在线窗口建立在线连接

如果在编程设备/PC上，已在项目中组态了可编程控制器，则可选用该方法。使用菜单命令**视图 > 在线**，打开SIMATIC管理器的在线窗口。它将显示可编程控制器上的项目数据(与此相对照的是，离线窗口将显示编程设备/PC的项目数据)。在线窗口同时显示S7程序和M7程序的可编程控制器的数据。

可以使用该项目视图中的功能来访问可编程控制器。SIMATIC管理器“PLC”菜单中的某些功能可以在在线窗口中激活，但不能在离线窗口中激活。

有下列两种访问类型：

- **通过已组态的硬件访问**
  这意味着您只能访问离线组态的模块。可以访问哪些在线模块取决于可编程模块组态时的MPI地址设置。
- **不通过已组态的硬件进行的访问**
  这要求存在着独立于硬件而创建的S7程序或M7程序。(即，它直接位于项目之下)。在此可以通过指定S7/M7程序对象属性中相应的MPI地址，来决定哪些在线模块可以访问。

在线窗口的访问组合了可编程控制系统的数据和编程设备的相关数据。例如，如果在线打开项目下的S7块，显示的内容由以下部分构成：

- 来自S7可编程控制器中CPU的块的代码段
- 注释和符号，来自编程设备的数据库(假如它们是离线存在的)。当您直接从不存在项目结构的连接着的CPU上打开块时，它们将以CPU中的存在状态显示，即不带符号和注释。

18.1.3 在多项目中在线访问PLC

使用所分配的PG/PC进行的项目间访问

用于对象“PG/PC”和“SIMATIC PC站”的“分配PG/PC”功能也可用于多项目。

您可在多项目的任意项目中指定进行在线访问的目标模块。该过程与您只使用一个项目进行工作时的过程相同。

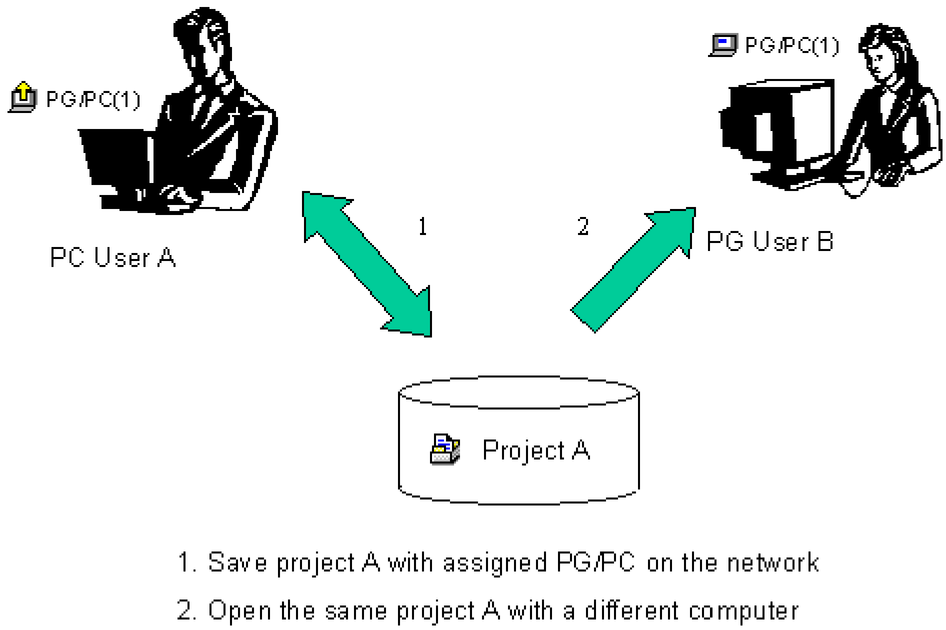
要求

- 您用于访问PLC的PG/PC或PC站必须已经在多项目中的任何一个项目中进行了分配。
  注意：当打开对应的项目时，所分配的PG/PC或PC站将用高亮度黄色显示。
  只有在打开项目的PG已正确分配时，PG/PC的分配才可见。
- 链接了所有项目的子网。
- 多项目的所有项目已经编译完毕，且组态数据已下载给参与的站；例如，向所有参与建立PG/PC与目标模块之间连接的模块提供路由信息。
- 目标模块可通过网络进行访问。

使用分布式项目进行工作时的可能问题

如果项目的分配发生改变，且项目在一个没有创建该项目的PG/PC上打开，则PG/PC分配不可见。
不过，所组态的PG/PC对象将仍然保持“已分配”状态，但具有“错误”的PG/PC。

此时，您必须清除现有的分配，并重新分配PG/PC对象。这时，可毫无问题地在多项目内对模块进行在线访问。



使用分布式项目进行工作的提示

如果有一个以上的团队成员希望访问与他们的PG联机的PLC，有用的方法是，在多项目中创建一个“PG/PC”或“SIMATIC PC站”对象，然后为PG的每个站进行分配。

根据打开了项目的PG，SIMATIC管理器将用黄色箭头指示且只指示已分配给该PG的对象。

18.1.4 用于访问可编程控制器的密码保护

使用口令保护，可以：

- 保护CPU中的用户程序，防止未授权的修改(写保护)
- 保护用户程序的编程知识产权(读保护)
- 防止将会干涉进程的在线功能

只有模块支持时，才能使用密码来保护模块或 MMC (例如，对于 CPU 31xC)的内容。

如果要使用密码来保护模块或 MMC 的内容，必须在分配模块参数时，定义保护级别和设置密码，然后将修改后的参数下载到模块。

如果在执行在线功能或访问 MMC 内容时需要密码，将会出现“请输入密码”对话框。输入正确的密码后，将获得模块的访问权限，对应着参数分配时所设定的特定的保护级别。然后，就可以为受保护的模块建立在线连接，执行属于该保护级别的在线功能。

使用菜单命令 **PLC > 访问权限设置**，可以直接调用“请输入密码”对话框。这样，例如在会话开始输入一次密码，以后的在线访问就不会再询问密码了。密码将一直有效，直到关闭 SIMATIC 管理器或使用菜单命令 **PLC > 访问权限 > 取消**将密码取消。

CPU参数	**说明**
测试操作/过程操作 (不适用于S7-400或CPU 318-2)	可以在“保护”标签中设置。 在过程操作时，诸如程序状态或监视/修改变量这些测试功能将受到限制，以便不会超出设置的扫描周期的允许增量。也就是说，例如，在程序状态中不允许使用调用条件，以及在编程的循环中状态显示将在返回点中断。 测试时使用断点，在过程操作时不能使用单步程序执行。 在测试操作时，可以毫无限制地通过编程设备/PC使用所有的测试功能，即使它们会导致扫描周期地大幅提高。
防护等级	可以在“保护”标签中设置。根据所知的正确密码，可以进行CPU的写或读/写访问。在此标签中设置密码。

18.1.5 更新窗口的内容

请注意以下事项：

- 由于用户操作而导致项目的在线窗口改变(例如，下载或删除块)不会在任何打开的“可访问节点”窗口中更新。
- 任何在“可访问节点”窗口的改变也不会在项目任何打开的在线窗口中自动改变。

要更新并行打开的窗口中的显示，必须在该窗口中明确地刷新显示(使用菜单命令或功能键F5)。

18.2 显示和改变工作模式

例如，通过该功能，可以在更正错误之后，将 CPU 重新切换到运行模式。

显示工作模式

1. 打开项目，然后选择一个S7/M7程序，或使用菜单命令**PLC > 显示可访问节点**来打开“可访问节点”窗口，然后选择一个节点(“MPI=...”)。
2. 选择菜单命令PLC > 诊断/设置 > 工作模式。

该对话框显示当前和最近一次工作模式，以及模块上模式选择器的当前设置。对于不能显示当前按键开关设置的模块，显示文本“未定义”。

改变工作模式

可使用按钮来改变CPU的模式。只有可以在当前工作模式中选择的按钮才是激活的。

18.3 显示和设置时间与日期

18.3.1 具有时区设置和夏令/冬令时的CPU时钟

使用 STEP 7 V5.1，Service Pack 2，除了日/日期的时间，在新的 CPU (固件 V3 或更高版本)中也可以组态或估算下列分配：

- 夏令/冬令时
- 显示时区的偏移量因数

显示时区

用全局的、连续和没有中断的 TOD，即模块时间进行系统操作。

本地自动化系统允许计算当地时间，它与模块时间不同，并且可以被用户程序使用。当地时间不直接输入，而是使用模块时间加/减与模块时间的时间差来计算。

夏令/冬令时

当设置TOD和日期时，也可以设置夏令或冬令时。例如，当从夏令时切换到冬令时，每个用户程序只考虑对于模块时间的时间差。可以用从英特网上获得的块来实现此改变。

读和调节TOD和TOD状态

夏令/冬令时标识符和对于模块时间的时间差包括在日的时间(TOD)状态中。

有下列选项可用于读或调节TOD及其状态：

使用STEP 7 (在线)

- 通过菜单命令 **PLC > 诊断/设置 > 调节TOD** (读和调节)
- 通过“模块信息”对话框，“时间系统”标签(只读)

在用户程序中

- SFC 100“SET_CLKS”(读和调节)
- SFC 51“RDSYSST”，具有SZL 132，索引8 (只读)

诊断缓冲区、消息和OB起始信息中的时间标志

时间标记使用模块时间生成

TOD中断

当冬令时切换为夏令时的时候，如果由于“时间跳转”没有触发 TOD 中断，OB 80 被调用。

对于夏令/冬令时转换，TOD 中断周期用分钟和小时周期维持。

TOD同步

组态为 TOD 主站(例如，在 CPU 寄存器“诊断/时钟”中)的 CPU 始终与其它具有模块时间和当前 TOD 状态的时钟同步。

18.4 更新固化程序

18.4.1 在线更新模块和子模块中的固化程序

从 STEP 7 V5.1 Service Pack 3 起，您可采取标准化的方法，对站中的模块或子模块进行在线更新。为此，可如下所述进行操作：

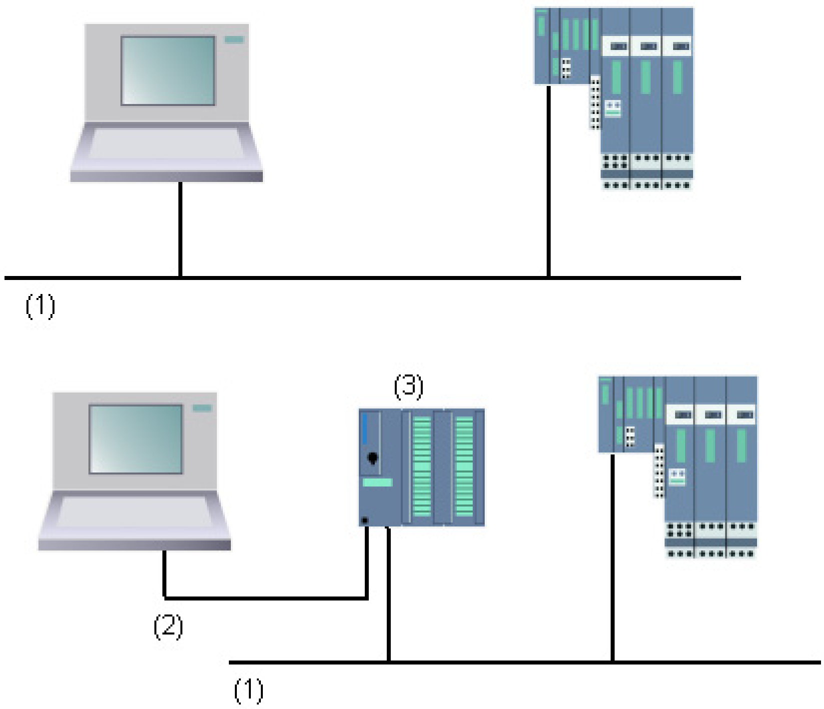
原则

为更新模块(例如CPU、CP或IM)上的固化程序，您必须获取包含有最新固化程序的文件(*.UPD)。
从这些文件选择一个，并将其下载给模块(PLC菜单)。

前提条件

将要对其固化程序进行更新的站点模块必须是在线的，例如当将编程设备(PG)连接到与要更新其固化程序的模块相同的PROFIBUS时。当将编程设备(PG)连接到DP主站CPU的MPI接口，且要更新其固化程序的模块将连接到DP接口的PROFIBUS时，也可对固化程序进行更新。CPU必须支持MPI接口和DP接口之间的S7路由。
模块自身必须支持固化程序更新。

在PG/PC的文件系统中必须存在包含最新固化程序版本的文件。只有**同一个**固化程序版本的文件才能位于同一个文件夹。



(1) PROFIBUS子网

(2) MPI子网

(3) 具有MPI接口和DP接口的CPU (具有S7路由选择)

HW组态的步骤

1. 打开含有要更新模块的站。
2. 选择模块
   对于PROFIBUS DP接口模块，例如IM 151，可选择DP从站图标。此时，它就是代表ET 200S的模块。
3. 选择菜单命令**PLC > 更新固化程序**。
   如果所选模块或所选DP从站支持"更新固化程序"功能，则您只可激活菜单命令。
4. 在所显示的“更新固化程序”对话框中，单击“浏览”按钮，并选择固化程序更新文件(*.UPD)的路径。
5. 在选择文件之后，“更新固化程序”对话框的下部域将包含有许多信息，这些信息将告诉您文件适合于哪些模块，以及适用于哪些固化程序版本。
6. 单击“运行”按钮。
   STEP 7将检查所选文件可由模块进行解释。如果检查结果为肯定的，则将文件下载给模块。
   如果需要修改CPU的工作模式，则将出现一个对话框提示您执行这些步骤。
   模块随后将独立执行固化程序更新。
   注意：对于固化程序更新，例如CPU 317-2PN/DP，通常将建立一个与CPU的

单独连接。在这种情况下，过程将可能中断。如果没有任何资源可供另一个连接使用，那么，将自动取代已经存在的连接。在这种情况下，连接将无法中断。传送对话框中的“取消”按钮将变为灰色，不能使用。

7. 在STEP 7中，检查(读取CPU诊断缓冲区)模块是否能够使用新的固化程序启动。

运行期间更新固化程序的影响

可以通过更新对话框中的选项，来决定在更新之后，立即激活新的固化程序。

如果您选择该选项，站点将执行类似于断电再重新上电的重启动。结果，将可能发生CPU仍然处于STOP模式或用户程序的处理受到不利的影响。您需要对运行的设备采取适当的预防措施来预先考虑和调解这些状况。
例如，在重启动期间，站点的所有模块均出现故障，包括现有的F I/O。
F I/O在掉电期间输出一个通讯错误给接口，然后安全关闭 - 它已被传递。传递的该信息没有被重新启动的接口清除。您必须分别取消模块的传递。然而，其结果是，与安全相关的应用程序将不能运行。

19 下载和上传

19.1 从PG/PC下载至可编程控制器

19.1.1 下载要求

下载到可编程控制器的要求

- 编程设备和可编程控制器中的CPU之间必须存在一个连接(例如，通过多点接口)。
- 必须可以访问可编程控制器。
- 为将块下载给PLC，在项目的对象属性对话框中必须已经为“使用”选择了条目“STEP 7”。
- 您正在下载的程序已经完成了编译，且没有任何错误。
- CPU必须处于允许进行下载的工作模式下(STOP或RUN-P)。
  请注意，在RUN-P模式下，程序将每次下载给一个块。如果通过这样来覆盖的旧CPU程序，则可能会导致冲突，例如，当块参数已经改变时。CPU在处理该循环时将转为STOP模式。因此，我们建议您在下载之前将CPU切换到STOP模式。
- 如果离线打开块，并希望对其进行下载，则CPU必须链接到SIMATIC管理器中的在线用户程序。
- 在您下载用户程序之前，您应复位CPU，以确保CPU上没有任何“旧的”块。

STOP模式

在进行下列操作之前，将工作模式从RUN设置为STOP：

- 下载完整用户程序或其中的一部分给 CPU
- 执行 CPU 的存储器复位
- 压缩用户存储器

重新启动(热启动(转换到RUN模式))

如果在“STOP”模式下执行了重启动(暖启动)，则程序将重新启动，并首先处理处于 STARTUP 模式下的启动程序(位于块 OB100 中)。如果启动成功，则 CPU 切换为 RUN 模式。在下列情况下将需要重启动(暖启动)：

- 复位 CPU
- 在 STOP 模式下下载用户程序

19.1.2 保存块和下载块之间的差别

应该始终区分开来保存块和下载块。

	保存	下载
菜单命令	**文件 > 保存** **文件 > 另存为**	**PLC > 下载**
功能	在编辑器中块的当前状态保存在编程设备的硬盘上。	编辑器中块的当前状态只下载到CPU。
语法检查	语法检查运行。任何错误都在对话框中报告。另外，也显示出错的原因和出错的位置。在保存或下载块之前，必须更正这些错误。如果没有发现语法错误，则将块编译成机器代码，并可以保存或下载块。	语法检查运行。任何错误都在对话框中报告。另外，也显示出错的原因和出错的位置。在保存或下载块之前，必须更正这些错误。如果没有发现语法错误，则将块编译成机器代码，并可以保存或下载块。

不管是在线还是离线打开块，表的应用总是独立的。

块改变的提示 - 先保存然后下载

在声明表中输入新创建的块或逻辑块代码段中的改变，或在数据块中输入新的或改变的数据值，必须保存相关的块。在编辑器中进行的任何改变，使用菜单命令 **PLC > 下载**在编辑器中进行任何改变或将其传送到CPU，例如，用于测试小的改变，这些改变在退出编辑器前也必须保存到编程设备的硬盘上。否则，将在CPU中和编程设备上有不同版本的用户程序。通常建议首先保存所有改变，然后再下载它们。

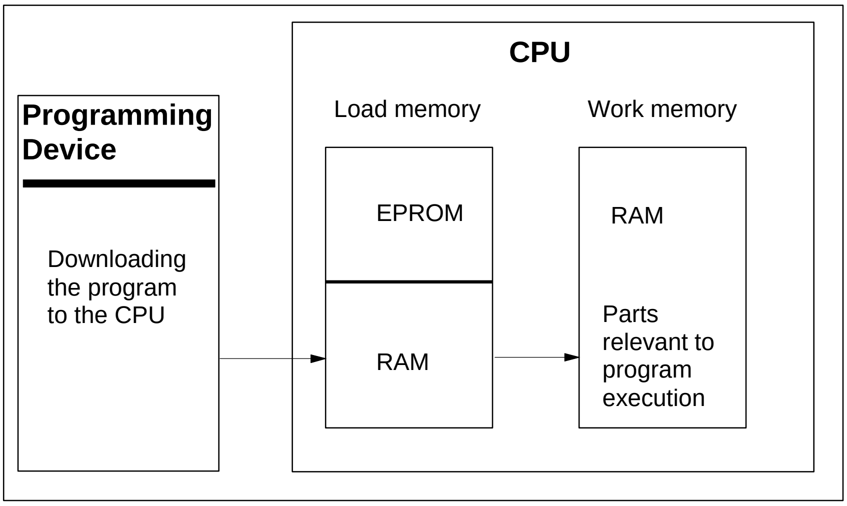
19.1.3 CPU中的装入存储器和工作存储器

在完成组态、参数分配和程序创建并建立在线连接后，可以将完整的用户程序或各个块下载至可编程控制器。要测试各个块，必须下载至少一个组织块(OB)、功能块(FB)、OB 中调用的功能(FC)以及使用的数据块(DB)。要下载组态硬件时所创建的系统数据、已组态的网络以及创建到可编程控制器的连接表时，请下载“系统数据”对象。

使用 SIMATIC 管理器将用户程序下载到可编程控制器(例如在程序测试的结束阶段)，或运行已完成的用户程序。

关系 - 装入存储器和工作存储器

完整的用户程序下载到装入存储器；与程序执行相关的部分也装入到工作存储器中。



CPU装入存储器

- 装入存储器用于存储没有符号表的用户程序和注释(这些仍保留在编程设备的存储器中)。
- 未标记为启动时所需要的块将只存储在装入存储器中。
- 根据可编程控制器，装入存储器可以是 RAM、ROM 或 EPROM 存储器。
- 装入存储器也可以具有集成的 EEPROM 部分，以及集成的 RAM 部分(例如 CPU 312 IFM 和 CPU 314 IFM)。
- 在 S7-400 中，必须使用存储卡(RAM 或 EEPROM)以扩展装入存储器。

CPU工作存储器

工作存储器(集成的 RAM)用于存储程序处理所需要的用户程序的部分。

可能的下载/上传过程

- 使用下载功能将用户程序或可装入对象(例如块)下载到可编程控制器。如果块已存在于 CPU 的 RAM 中，将提示您确认是否要覆盖块。
- 可以在项目窗口中选择可加载的对象，并通过 SIMATIC 管理器(菜单命令：**PLC > 下载**)。
- 在编写块、组态硬件和网络时，可以通过正在使用的应用程序主窗口中的菜单(菜单命令：**PLC > 下载**)直接下载当前正在编辑的项目。
- 也可以打开具有可编程控制器视图的在线窗口(例如，使用**视图 > 在线** 或 **PLC > 显示可访问节点**)，将想要下载的对象复制到在线窗口。

另外，可以通过下载功能将块的当前内容从 CPU 的 RAM 装入存储器上传到您的编程设备。

19.1.4 依赖于装入存储器的下载方法

CPU 的装入存储器划分为 RAM 和 EEPROM 区域决定了下载用户程序或用户程序中的块的可用方法。下列方法可用于下载数据到 CPU：

装入存储器	装载方法	PG与PLC之间的通讯类型
RAM	下载和删除各个块	在线PG - PLC连接
	下载和删除完整的用户程序	在线PG - PLC连接
	重新装入各个块	在线PG - PLC连接
集成的(仅适用于S7-300)或插入式EPROM	下载完整的用户程序	在线PG - PLC连接
插入式EPROM	下载完整的用户程序	外部装入EPROM并插入存储卡，或通过在插入EPROM的PLC上的在线连接。

通过在线连接下载到RAM

如果出现掉电并且 RAM 未备份，则可编程控制器中的数据会丢失。在这种情况下，RAM 中的数据将会丢失。

保存到EPROM存储卡

块或用户程序保存在 EPROM 存储卡中，然后存储卡插入 CPU 的插槽。

存储卡是便携式的数据介质。它们由编程设备写入，然后插入 CPU 相应的槽中。

在关闭电源后和 CPU 复位时，存储卡上存储的数据仍保留。如果 RAM 没有备份，在 CPU 的存储器复位和电源关闭后，当电源恢复时 EPROM 的内容再次复制到 CPU 存储器的 RAM 区域。

保存在集成的EPROM中

对于 CPU 312，也可以将 RAM 的内容保存到集成的 EPROM 中。在电源关闭期间，在集成的 EPROM 中的数据仍保留。如果 RAM 没有备份，在 CPU 的存储器复位和电源关闭后，当电源恢复时集成的 EPROM 的内容再次复制到 CPU 存储器的 RAM 区域。

19.1.5 将程序下载到S7 CPU

19.1.5.1 通过项目管理下载

1. 在项目窗口中，选择想要下载的用户程序或块。
2. 通过选择菜单命令**PLC > 下载**将所选对象下载到可编程逻辑控制器。

其它方法(拖放)

3. 打开项目的离线窗口和在线窗口。
4. 在离线窗口中选择想要下载的对象，并将它们拖到在线窗口中。

19.1.5.2 不通过项目管理而进行的下载

1. 使用菜单命令**PLC > 显示可访问节点**或通过点击工具栏中的相应按钮，打开“可访问节点”窗口。
2. 在“可访问节点”窗口中双击所需节点(“MPI=...”)以显示“块”文件夹。
3. 打开想要将其用户程序或块下载到可编程逻辑控制器的库或项目。为此，使用菜单命令**文件 > 打开**。
4. 在为项目或库打开的窗口中，选择想要下载的对象。
5. 通过在“可访问节点”窗口中使用拖放将对象复制到“块”文件夹，可将对象下载到可编程逻辑控制器。

19.1.5.3 在可编程控制器中重新加载块

可以用新版本的块覆盖已存在于 S7 可编程逻辑控制器 CPU 中的装入存储器(RAM)或工作存储器中的块(重载它们)。覆盖已存在的版本。

重载 S7 块的步骤与下载相同。将显示提示信息，询问是否希望覆盖已存在的块。

不能删除存储在 EPROM 中的块，但是一旦重载将会声明无效。装入替换的块到 RAM 中。这在装入存储器或工作存储器中造成间隙。如果这些间隙最后意味着没有新的块可以下载，则应该压缩存储器。

注意

如果电源先关闭然后再打开，并且RAM没有电池装备，或随后CPU的存储器复位，“旧”的块再次有效。

19.1.5.4 在集成的EPROM上保存已下载的块

对于带一个集成 EPROM 的 CPU (例如 CPU 312)，可将 RAM 中的块保存到集成 EPROM 中，从而在发生断电或存储器复位时不会丢失数据。

1. 使用菜单命令**视图 > 在线**，显示含已打开项目在线视图的窗口，或，在工具栏中点击“可访问节点”按钮或选择菜单命令**PLC > 显示可访问节点**，打开“可访问节点”窗口。
2. 在项目的在线窗口中选择S7或M7程序，或在“可访问节点”窗口中选择节点。
3. 使用下列方法之一，在CPU上选择要保存的“块”文件夹：
   - 如果正在使用项目管理，那么在项目的在线窗口中
   - 如果没有使用项目管理，那么在“可访问节点”窗口中
4. 选择菜单命令**PLC > 保存RAM到ROM**。

19.1.5.5 通过EPROM存储器卡进行下载

要求

要访问专为 S7 可编程逻辑控制器而设计的编程设备中的 EPROM 存储卡，需要合适的 EPROM 驱动程序。要访问为 M7 可编程控制系统设计的编程设备中的 EPROM 存储卡，必须安装 FLASH 文件系统(仅适用于 PG 720、PG 740 和 PG 760)。当安装 STEP 7 标准软件包时，EPROM 驱动程序和 Flash 文件系统作为选件提供。如果使用 PC，则需要外部编程器来保存到 EPROM 存储卡。

也可以在以后安装驱动程序。为此，通过**开始 > Simatic > STEP 7 > 存储卡参数分配**或通过控制面板(双击“存储卡参数分配”图标)调用相应的对话框。

保存在存储卡上

要将块或用户程序保存到存储卡，请执行如下操作：

1. 在编程设备的插槽中插入存储卡。
2. 通过下列方式打开“存储卡”窗口：
   - 点击工具栏中的“存储卡”按钮。如果必要，使用菜单命令**视图 > 工具栏**激活工具栏。
   - 也可以选择菜单命令**文件 > S7 存储卡** > **打开**。
3. 打开或激活下列窗口之一，以显示想要保存的块：可以有下列窗口：
   - 项目窗口，“在线”视图
   - 项目窗口，“离线”视图
   - 库窗口
   - “可访问节点”窗口
4. 选择“块”文件夹或各个块，并将它们复制到“S7存储卡”窗口。
5. 如果块已存在于存储卡上，会显示出错消息。这种情况下，擦除存储卡的内容，并重复从步骤2开始的步骤。

19.2 从PG编译并下载几个对象

19.2.1 关于下载的要求和注意事项

下载块文件夹

对于块文件夹，只能下载逻辑块。块文件夹中的其他对象，例如系统数据(SDB)等，均不能在这里下载。SDB 将通过“硬件”对象进行下载。

注意

对于PCS 7项目，正如不能在SIMATIC管理器中对其进行下载一样，使用对话框“编译和下载对象”也无法对块进行下载。对于PCS 7项目，有以下限制：PLC只能通过CFC进行下载，以便确保下载期间的正确次序。这一操作必须完成，以避免CPU转到STOP模式。

为确定给定的项目是否为PCS 7项目，请检查项目属性。

下载故障保护控制器的F共享

出于安全原因，在下载已修改的 F-共享之前必须输入密码。由于这一原因，使用“编译和下载对象”功能下，下载过程将由于出现错误消息而中止。在这种情况下，将与选项包一起，把相应的程序部分装载到 PLC。。

下载硬件组态

只有在不会触发任何错误消息或提示的情况下，才能通过“编译和下载对象”功能，对所有选择的对象无中断地下载硬件组态(例如下载离线SDB)。下面的章节提供了有关如何避免出现这样的消息或提示的信息。

硬件组态的下载要求

- CPU 必须位于 STOP 模式。
- 必须可以建立与 CPU 的在线连接。对于所选 CPU 或所选块文件夹而言，在运行“编译和下载对象”功能之前，受密码保护的 CPU 将需要一个经授权的连接或输入密码(“编辑”按钮)。

- 对下载正在使用的目标系统的接口，不能进行任何重新组态：
  - 禁止修改接口地址。
  - 如果您改变了网络设置，这可能意味着不是所有的模块都能访问。
- 对于 H-CPU 而言，在运行“编译和下载对象”功能(选择“CPU”对象并点击“编辑”按钮)之前，您可选择接收下载的 CPU (H-CPU 0 或 H-CPU 1)。
- 禁止修改下面的 CPU 参数：
  - CPU 上的局部数据和通讯资源的最大值(“存储器”标签)
  - F-CPU 的密码保护(“保护”标签)
- 对于各个已组态的模块，下面的条件必须满足：
  - 所组态模块的订购号必须与实际插入模块的订购号完全相同。
  - 所组态模块的硬件版本不能高于实际插入模块的硬件版本。
  - 站名称、模块名称以及设备名称在上一次下载之后没有进行修改。然而，您可分配一个新的设备名称。

关于下载过程的提示

- 所有离线的 SDB 均将下载(也就是说，除了硬件组态以外，也将下载连接 SDB 以及通过全局数据组态创建的 SDB)。
- 只有在先前的编译期间没有发生任何错误时，才能执行下载。
- 在下载期间，任何错误反馈消息都将隐藏。例如，如果达到 CPU 存储器容量，则自动压缩数据，而不会通知用户。
- 在下载完成之后，所下载的模块将处于 STOP 模式(除了那些不通知用户就自动停止并重新启动的模块之外)。

提示

如果在下载完成之后，出现一条消息说对象的下载已完成并有警告，那么，请务必浏览日志文件的内容。这既可能是对象尚未下载，也可能是对象没有完全下载。

19.2.2 如何编译和下载对象

在“编译和下载对象”对话框中，您可准备一些对象，这些对象可在项目或多项目中选择，用于传送给 PLC 及其后续的下载(如果希望的话)。该对话框可用于站、项目或多项目中的对象。

根据所选择的对象的不同，某些信息可能不显示。此外，下面所描述的功能并非都可供这些对象使用。这些限制尤其可能适用于那些使用选项软件包创建的对象。

对于块文件夹中的块，“编译”意味着检查块的一致性。为了便于描述，下文中把块的一致性检查称为编译。

过程：

1. 在SIMATIC管理器中，选择您希望编译或编译并下载的对象。在SIMATIC管理器中可选择下列对象：
   - 多项目
   - 项目
   - 站
   - 没有站分配的 S7 程序
2. 在SIMATIC管理器中，选择菜单命令**PLC > 编译和下载对象**。
3. 如果您希望执行块的检查而不将块下载给PLC，可选择“仅编译”。如果不希望将这些对象中的任何一个下载给PLC，可选择该选项。
4. 为避免由于编译错误而导致站的不完整下载，可选择“出现编译错误时不进行任何下载”复选框。如果选择该复选框，则不下载任何东西。如果没有选择复选框，则编译无误的所有对象均下载。导致编译期间出现错误的对象不下载。
5. 如果还希望编译和下载连接，可选择“考虑连接”复选框。

   如果希望为项目或多项目下载或编译所组态的连接，则选择该复选框。只有当项目或多项目被选作作为启动点使用的对象时才可使用该复选框。

   多项目特别适合用作启动点，因为项目间连接的所有连接伙伴也可从该对象中下载。

   如果选择该复选框，则所有的“硬件”和“连接”对象都将自动显示，包括编译过程期间(不能改变设置)。如果清除“仅编译”复选框，则所有的“硬件”和“连接”对象均将自动显示，包括下载期间(不能改变设置)。

   如果清除“考虑连接”复选框，只要不同时关闭对话框，则STEP 7将恢复“编译”和“下载”列的原来设置。

注意：当使用本对话框下载连接时，硬件也总是下载。因此，只有在CPU处于STOP模式时才能进行下载。使用NetPro，您可下载单个的连接。

6. 在“编译”和“下载”列中，选择希望编译或下载的对象。您的选择会有复选标记进行标识。如果在步骤3中选择“仅编译”，则“下载”列将变为灰色，且不能使用。
7. 点击“启动”开始编译。
8. 按照屏幕上的说明进行操作。

一旦完成该操作，可点击“全部”按钮以便查看整个动作的记录，或点击“单个对象”以便只查看从对象表中选择的对象的记录。

19.3 从可编程控制器上传至PG/PC

19.3.1 从可编程控制器上传至PG/PC

该功能支持执行下列动作：

- 保存来自可编程控制器的信息(例如，用于服务目的)
- 如果在开始组态之前就有硬件组件，可快速组态和编辑一个站。

保存来自可编程控制器的信息

有必要使用该措施的情况是，例如，在CPU上运行的版本的离线项目数据不可用或部分不可用时。此时，至少可以检索在线可用的项目数据，并将它们上传至编程设备。

快速组态

组态好硬件并重启动(暖启动)站后，如果已经将组态数据从可编程控制器上传至编程设备，那么更易于输入站组态。这为您提供了站组态和单个模块的类型。之后，所要做的就是更详细地指定这些模块(订货号)，并为它们分配参数。

将下列信息上传至编程设备：

- S7-300：用于中央机架和任何扩展机架的组态
- S7-400：带一个 CPU 的中央机架以及无扩展机架的信号模块的组态
- 分布式 I/O 的组态数据不能上传至编程设备。

如果在可编程控制器上没有组态信息，那么上传该信息；例如，在系统中执行存储器复位的情况下。否则，**上传**功能可提供更好的结果。

对于不带分布式I/O的S7-300系统，所需做的工作就是更详细地指定这些模块(订货号)，然后为它们分配参数。

注意

上传数据时(如果还没有离线组态)，STEP 7不能确定组件的所有订货号。

组态硬件时，可使用菜单命令**选项 > 指定模块**，输入“不完整”的订货号。通过该方式，可以将参数分配给STEP 7不能识别的模块(即，在“硬件目录”窗口中没有出现的模块)；然而，这时STEP 7不会检查是否遵守了参数规则。

从可编程控制器上传时的限制条件

下列限制条件适用于从可编程控制器上传至编程设备的数据：

- 块不包含参数、变量和标签的任何符号名
- 块不包含任何注释
- 所有系统数据会随整个程序一同上传，系统只能继续处理属于“组态硬件”应用的系统数据
- 不能更进一步处理用于全局数据通讯(GD)和组态与符号有关的消息的数据
- 强制作业不随其它数据一起上传至编程设备。它们必须单独保存为变量表(VAT)
- 模块对话框中的注释不上传
- 只有在组态期间选择了相应选项时才显示模块的名称(HW Config：**选项 > 自定义**下的对话框中的“在可编程逻辑控制器中保存对象名称”选项)。

19.3.2 上传站

使用菜单命令 **PLC > 上传工作站**，可以将当前组态和所有块从所选的可编程控制器上传到编程设备。

为此，STEP 7 在将要保存组态的当前项目中创建新的工作站。可以改变新工作站的预设名(例如，“SIMATIC 300-Station(1)”)。插入的站将在在线视图和离线视图中都显示。

当打开项目时，可以选择菜单命令。在项目窗口或视图(在线或离线)中选择对象将不会对菜单命令造成影响。

可以使用此功能简化组态操作。

- 对于S7-300可编程控制器，上传实际硬件配置的组态(包括扩展机架)，但没有分布式I/O (DP)。
- 对于S7-400可编程控制器，上传机架配置，但没有扩展机架和分布式的I/O。

对于不带分布式I/O的S7-300系统，必须更为详细地指定模块(订货号)并为它们分配参数。

上传工作站时的限制

上传到编程设备的数据有下列限制：

- 块不包含参数、变量和标签的任何符号名
- 块不包含任何注释
- 整个程序连同所有系统数据上传，因此不是所有数据均可进一步处理
- 不能进一步处理用于全局数据通讯(GD)、组态符号相关的消息和组态网络的数据
- 强制作业不能上传到编程设备，于是加载回可编程控制器。

19.3.3 从S7 CPU上传块

可以使用SIMATIC管理器将S7块从CPU上传到编程设备的硬盘。在下列情况下将块上传到编程设备是十分有用的：

- 制作CPU中加载的当前用户程序的备份副本。例如，在维修后或在由维护人员对CPU进行存储器复位后，可再次下载该备份。
- 可以将用户程序从CPU上传到编程设备，并在编程设备上对其进行编辑，例如用于故障诊断目的。这种情况下，不能访问程序文档的符号或注释。因此，我们建议此步骤仅适用于维护目的。

19.3.4 在PG/PC中编辑上传的块

19.3.4.1 在PG/PC中编辑上传的块

能够将块从CPU上传到编程设备具有下列用途：

- 在测试阶段，可以在CPU上直接更正块并对结果进行归档。
- 可以通过加载功能，将块的当前内容从CPU的RAM装入存储器上传到编程设备。

注意

在线和离线工作时的时间标志冲突

下列步骤会导致时间标志冲突，因此必须避免。

下列情况下在线打开块时会引起时间标志冲突：

- 在线所作的改变未在离线S7用户程序中保存
- 离线所作的改变未下载到CPU

下列情况下在线打开块时会引起时间标志冲突：

- 将具有时间标志冲突的在线块离线复制到S7用户程序，然后离线打开块。

两种不同的情况

当从CPU上传块到编程设备时，记住有两种不同的情况：

1. 块所属的用户程序在编程设备上。

2. 块所属的用户程序不在编程设备上。

这意味着下面列出的不能下载到CPU的程序部分是不可用的。这些组件是：

- 其符号名具有地址和注释的符号表
- 梯形图或功能块图程序的程序段注释
- 语句表程序的行注释
- 用户自定义的数据类型

19.3.4.2 如果用户程序在PG/PC上编辑上传的块

要从CPU编辑块，可如下操作：

1. 在SIMATIC管理器中打开项目的在线窗口。
2. 从在线窗口中选择“块”文件夹。将显示加载的块列表。
3. 现在选择块，将其打开并进行编辑。
4. 选择菜单命令**文件 > 保存**以在编程设备上离线保存改变。
5. 选择菜单命令PLC > **下载**以将改变的块下载到可编程控制器。

19.3.4.3 如果用户程序不在PG/PC上编辑上传的块

要从CPU编辑块，可如下操作：

1. 在SIMATIC管理器中，点击“可访问节点”工具栏按钮或选择菜单命令**PLC > 显示可访问节点**。
2. 从显示的列表中选择节点(“MPI=...”对象)，然后打开“块”文件夹以显示块。
3. 现在可以打开块，并根据需要对其编辑、监视或复制。
4. 选择菜单命令**文件 > 另存为**，然后在对话框中为编程设备输入想要存储块的路径。
5. 选择菜单命令**PLC > 下载**以将改变的块下载到可编程控制器。

19.4 在可编程控制器上删除

19.4.1 删除装载/工作存储器，并复位CPU

将用户程序下载到S7可编程控制器之前，应该在CPU上执行一次存储器复位，确保CPU上没有“旧”块。

存储器复位要求

CPU必须处于STOP模式，以执行存储器复位(模式选择器设为STOP，若为RUN-P，可通过菜单命令PLC > 诊断/设置 > 工作模式将模式改为STOP)。

在S7 CPU上执行存储器复位

在S7 CPU上执行存储器复位时，会执行如下各项：

- 复位CPU。
- 删除所有用户数据(块和系统数据块(SDB)，MPI 参数除外)。
- CPU 中断所有已存在的连接。
- 如果在EPROM (存储卡或集成EPROM)上存在数据，则在存储器复位后，CPU会将EPROM内容复制回存储器的RAM区。

保持诊断缓冲区的内容和MPI参数。

在M7 CPU/FM上执行存储器复位

在M7 CPU/FM上执行存储器复位时，执行如下各项：

- 恢复原始状态。
- 除MPI参数外，删除系统数据块(SDB)。
- CPU/FM终止所有已存在的连接。保持用户程序，当CPU从STOP切换到RUN之后，继续运行。

通过“存储器复位”功能，可在发生严重错误后恢复M7 CPU或FM的原始状态，方法是从工作存储器中删除当前系统数据块(SDB)，然后在只读存储器中重新装载SDB。在有些情况下，需要暖启动操作系统。为此，可使用模式选择器(切换到MRES位置)清除M7。只有在CPU/FM上使用RMOS32操作系统时，才能在SIMATIC M7 CPU或FM上使用模式选择器进行复位。

19.4.2 在可编程控制器上删除S7块

删除CPU上的单个块可能在CPU程序的测试阶段是必需的。块存储在CPU用户存储器的EPROM或RAM上(根据CPU和加载步骤)。

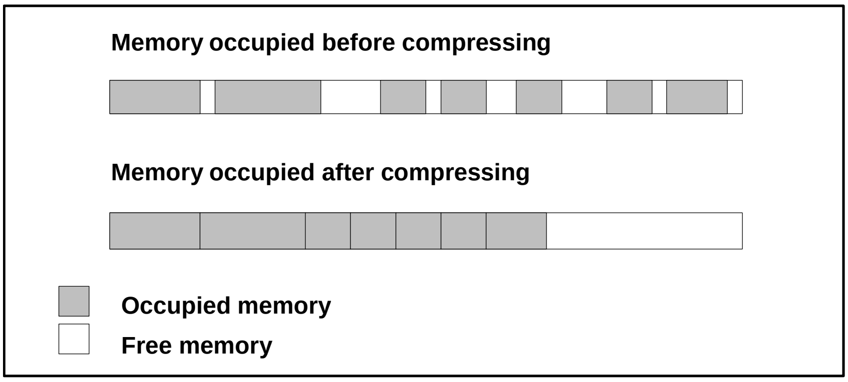
- 在RAM中的块可以直接删除。在装入或工作存储器中占用的空间将被释放，并可以再次使用。
- 在CPU的存储器复位后，在集成的EPROM中的块始终复制到RAM区域。在RAM中的副本可以直接删除。于是，删除的块在EPROM中标记为无效，直到下一次存储器复位或电源关闭而RAM没有备份。在CPU的存储器复位后，在集成的EPROM中的块始终复制到RAM区。可以直接删除RAM中的副本。在集成的EPROM中的块(例如，在CPU 312中)将由新的RAM内容覆盖，从而被删除。
- EPROM存储卡必须在编程设备中擦除。

19.5 压缩用户存储器(RAM)

19.5.1 用户存储器(RAM)中的间隔

在删除和重新加载块后，可能会在用户存储器(装入和工作存储器)中产生间隔，从而减少可使用的存储区域。使用压缩功能，可将现有块在用户存储器中无间隔地重新排列，并创建连续的空闲存储空间。

下图显示了存储器占用的块是如何通过压缩功能移位到一起的。



始终尝试在STOP模式中压缩存储器

只有在STOP模式下压缩存储器时，所有的间隔才靠拢。在RUN-P模式(模式选择器设置)中，因为当前正在处理的块是打开的，所以不能移位。压缩功能在RUN模式(模式选择器设置)下时不起作用(写保护！)。

19.5.2 压缩S7 CPU的存储内容

压缩存储器的方法

有两种方法可以压缩用户存储器，如下所述：

- 当下载到可编程控制器时如果没有足够的存储空间，将显示对话框通知出错。可以通过点击对话框中的相应按钮压缩存储器。
- 也可以采取预防性措施，显示存储器的利用情况(菜单命令**PLC > 诊断/设置 > 模块信息**，“存储器”标签页)，并根据需要启动压缩功能。

步骤

1. 在“可访问节点”窗口或项目的在线视图中选择S7程序
2. 选择菜单命令PLC > 诊断/设置 > 模块信息。
3. 在随后显示的对话框中选择“存储器”标签页。如果CPU支持压缩存储器功能，则在该标签页中存在一个相应功能的按钮。

20 调试

20.1 关于使用变量表进行测试的说明

变量表具有能够存储各种不同的测试环境的优点。从而可在操作期间或出于保养和维护的目的而很容易地进行测试和监控。对变量表的可存储数目没有任何限制。

当使用变量表进行测试时，下列功能都可使用：

- **监视变量**
  该功能将使您能够在可编程设备/PC上显示用户程序或CPU中单个变量的当前值。
- **修改变量**
  可使用该功能将固定值分配给用户程序或CPU的单个变量。在使用程序状态进行测试时也可以立即修改一次数值。
- **启用外围输出和激活修改值**
  这两个功能允许您将固定值分配给处于STOP模式的CPU的单个I/O输出。
- **强制变量**
  可使用该功能为用户程序或CPU的单个变量分配一个用户程序无法覆盖的固定值。

可设置或显示下列变量的值：

- 输入、输出、位存储器、定时器以及计数器
- 数据块的内容
- I/O (外围)

在变量表中输入您想要显示或修改的变量。

您可通过定义触发点和触发频率来确定何时以及每隔多久对变量进行监视或为其分配新的数值。

20.2 使用变量表进行监视和修改时的基本步骤

要要使用**监视**和**修改**功能，可如下操作：

1. 创建一个新的变量表或打开一个已经存在的变量表。
2. 编辑或检查变量表的内容。
3. 使用菜单命令**PLC > 连接到**，在当前变量表和所需的CPU之间建立在线连接。
4. 使用菜单命令**变量 > 触发器**，选择合适的触发点并设置触发频率。
5. 菜单命令**变量 > 监视**和**变量 > 修改**，打开、关闭监视和修改功能。
6. 使用菜单命令**表格 > 保存**或**表格 > 另存为**来保存所完成的变量表，以便可以随时再次调用它。

20.3 编辑并保存变量表

20.3.1 创建并打开变量表

在监视或修改变量之前，必须创建一个变量表(VAT)，并输入所要求的变量。可选择下列方法之一来创建变量表：

在SIMATIC管理器中：

- 选择“Blocks”文件夹，然后选择菜单命令**插入 > S7块 > 变量表**。在对话框中，可以给表一个名称(“符号名称”文本框)。双击该对象，可打开变量表。
- 选择一个连接，或在在线视图中，从可用的节点列表中选择S7或M7程序。使用菜单命令**PLC > 监视/修改变量**，创建一个未命名变量表。

在“监视/修改变量”中：

- 可使用菜单命令**表格 > 新建**来创建还没有分配给S7或M7程序的新变量表。可通过**表格> 打开**来打开已存在的表。
- 可使用工具栏中的相应符号来创建或打开变量表。

一旦创建了变量表，可以保存该变量表，打印输出，并反复用于监视和修改。

20.3.2 复制/移动变量表

可以在S7/M7程序的块文件夹中复制或移动变量表。

复制或移动变量表时，请注意：

- 将更新目标程序符号表中的现有符号。
- 当移动变量表时，来自源程序的符号表的相应符号也将移动到目标程序的符号表中。
- 当从块文件夹删除变量表时，也将删除来自S7/M7程序符号表的相应符号。
- 如果目标程序已包含具有相同名称的变量表，当复制变量表时，将分配下一个最高空闲编号。
- 如果目标程序已包含具有相同名称的变量表，可以在复制时重命名变量表(在缺省情况下，已存在的名称带有编号)。

20.3.3 保存变量表

当再次测试程序时，可以使用所保存的变量表来监视和修改变量。

1. 使用菜单命令**表格 > 保存**来保存变量表。
2. 如果已创建了变量表，那么现在必须给变量表指定一个名称，例如“ProgramTest_1”。

保存变量表时，将保存所有当前的设置和表格式。这意味着将保存“触发器”菜单项下所作的设置。

20.4 在变量表中输入变量

20.4.1 在变量表中插入地址或符号

选择要修改或监视其值的变量，然后在变量表中输入这些变量。从“外部”开始，“朝内”工作；这表示应该先选择输入，然后选择受输入影响以及影响输出的变量，最后选择输出。

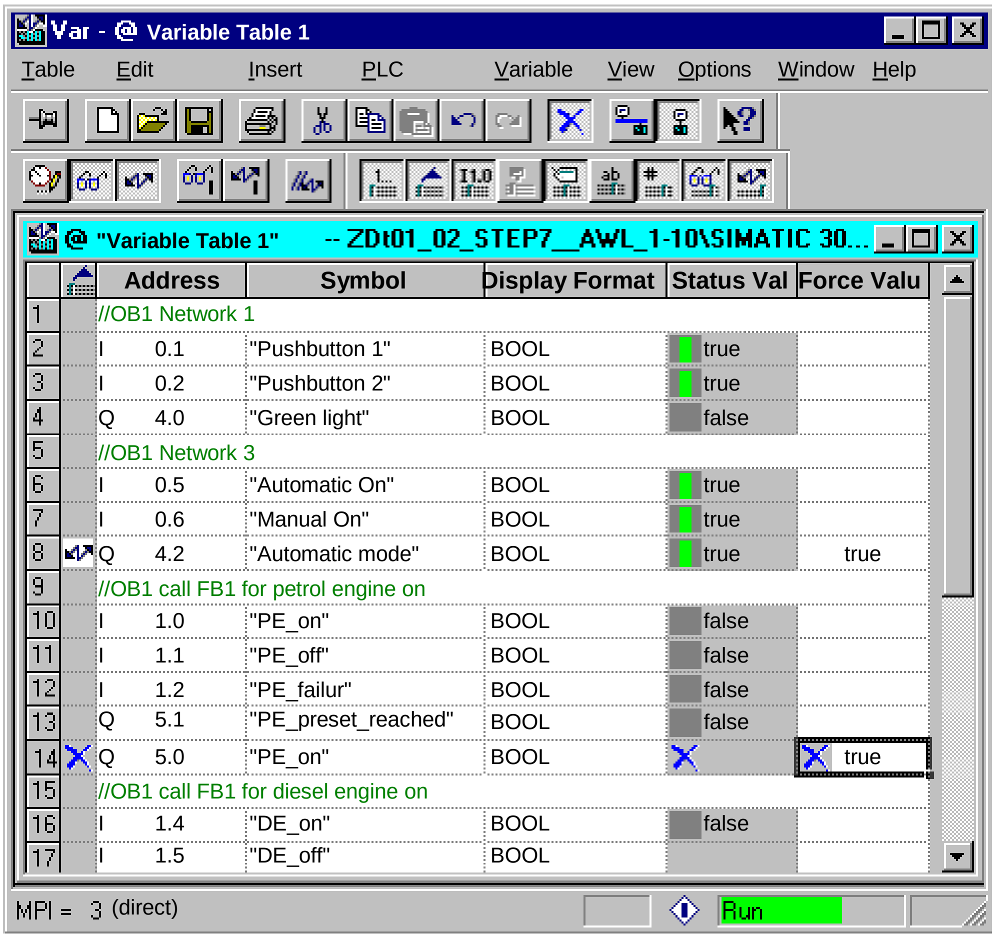
例如，如果希望监视输入位1.0、存储字5以及输出字节0，那么在“地址”列中按如下进行输入：

实例：

I 1.0
MW5
QB0

一个已完成的变量表的实例

下图所示为具有下列可见列的变量表：地址、符号、显示格式、监视值和修改值



	Address		Symbol	Display Format	Status Val	Force Valu
1	//OB1 Network 1					
2	I	0.1	"Pushbutton 1"	BOOL	true	
3	I	0.2	"Pushbutton 2"	BOOL	true	
4	Q	4.0	"Green light"	BOOL	false	
5	//OB1 Network 3					
6	I	0.5	"Automatic On"	BOOL	true	
7	I	0.6	"Manual On"	BOOL	true	
8	Q	4.2	"Automatic mode"	BOOL	true	true
9	//OB1 call FB1 for petrol engine on					
10	I	1.0	"PE_on"	BOOL	false	
11	I	1.1	"PE_off"	BOOL	false	
12	I	1.2	"PE_failur"	BOOL	false	
13	Q	5.1	"PE_preset_reached"	BOOL	false	
14	Q	5.0	"PE_on"	BOOL		true
15	//OB1 call FB1 for diesel engine on					
16	I	1.4	"DE_on"	BOOL	false	
17	I	1.5	"DE_off"	BOOL		

插入符号时的注意事项

- 输入希望通过地址修改、或作为符号的变量。可在“符号”列或“地址”列中输入符号和地址。然后该条目会在正确的列中自动写入。
  如果在符号表中定义了相应的符号，那么会自动填写符号列或地址列。
- 只能输入已经在符号表中定义过的那些符号。
- 必须完全按照符号表中的定义来输入符号。
- 含特殊字符的符号名称必须包含在引号内(例如，“Motor.Off”、“Motor+Off”、“Motor-Off”)。
- 为了在符号表中定义新的符号，可选择菜单命令**选项 > 符号表**。可从符号表中复制符号，并将其粘贴到变量表中。

语法检查

在变量表中输入变量时，在每行末尾执行语法检查。错误的条目以红色标记。
如果将光标放在用红色标记的行中，则会显示一条简要信息，告知错误原因。按下F1，可获取校正错误的注意事项。

注意

如果希望通过键盘编辑变量表(无鼠标)，那么应该启用“使用键盘时的简要信息”特性。

如有必要，可通过选择菜单命令**选项 > 自定义**，然后选择“常规”标签来改变变量表中的设置。

最大大小

在变量表中，每行最多可输入255个字符。不能通过回车键转到下一行。每个变量表最多可有1024行。这就是其最大大小。

20.4.2 在变量表中插入连续地址范围

1. 打开变量表。
2. 将光标放在希望连续地址范围插入其后的行的中间。
3. 选择菜单命令**插入 > 变量范围**。“变量的插入范围”对话框显示。
4. 在“从地址”域中输入地址作为开始地址。
5. 在“编号”域中输入要插入的行编号。
6. 从显示的列表选择要求的显示格式。
7. 点击“确定”按钮。

变量范围插入变量表中。

20.4.3 插入修改值

修改作为注释的值

如果希望制作无效变量的“修改值”，请使用**变量 > 修改作为注释的值**菜单命令。在要修改的变量值前的注释标识“//”表示这是无效的。也可以在“修改值”的前面插入命令标识“//”以代替菜单命令调用。可以通过再次调用**变量 > 修改作为注释的值**菜单命令或通过删除注释标识来反向设置“修改值”的无效性。

20.4.4 输入定时器上限

注意下列输入定时器的上限：

实例：W#16#3999 (BCD格式的最大值)

实例：

地址	监视格式	输入	修改值显示	解释
T1	SIMATIC_TIME	137	S5TIME#130MS	转变到毫秒
MW4	SIMATIC_TIME	137	S5TIME#890MS	BCD格式可能的表示法
MW4	HEX	137	W#16#0089	BCD格式可能的表示法
MW6	HEX	157	W#16#009D	不能以BCD格式表示，因此不能选择监视格式SIMATIC_TIME

注意

- 可以以毫秒步长输入定时器，但输入的值改变成时间帧。时间帧的长度依赖于输入时间值的长度(137变成130毫秒；7毫秒被取整)。
- 数据类型WORD的地址修改值(如IW1)转换成BCD格式。然而，不是每个位类型都是有效的BCD数字。如果输入不能表示为数据类型WORD的地址的SIMATIC_TIME，应用程序自动转换为缺省格式(此处：HEX，参见“选择监视格式”、“缺省命令(视图菜单)”)，以使输入的值可以显示。

SIMATIC_TIME格式中变量的BCD格式

SIMATIC_TIME格式中的变量值以BCD格式输入。
16个位具有如下意义：

| 0 0 x x | h h h h | t t t t | u u u u |

位15和14始终是零。
位13和12 (用xx标志)设置位0 - 11的乘数：
00 => 乘以10毫秒
01 => 乘以100毫秒
10 => 乘以1秒
11 => 乘以10秒
位11到8 数百(hhhh)
位7到4 数十(tttt)
位3到0 单位(uuuu)

20.4.5 输入计数器上限

注意下列输入计数器的上限：

计数器的上限： C#999
W#16#0999 (BCD格式的最大值)

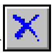
实例：

地址	监视格式	输入	修改值显示	解释
C1	COUNTER	137	C#137	转换
MW4	COUNTER	137	C#89	BCD格式可能的表示法
MW4	HEX	137	W#16#0089	BCD格式可能的表示法
MW6	HEX	157	W#16#009D	不能以BCD格式表示，因此不能选择监视格式COUNTER

注意

- 如果为计数器输入十进制的数字，而没有用C#标志值，此值自动地转换为BCD格式(137变成C#137)。
- 数据类型WORD的地址修改值(如IW1)转换成BCD格式。然而，不是每个位类型都是有效的BCD数字。如果输入不能表示为数据类型WORD地址的COUNTER，应用程序自动转换为缺省格式(此处：HEX，参见“选择监视格式”、“缺省命令(视图菜单)”)，以使输入的值可以显示。

20.4.6 插入注释行

注释行由注释标识“//”引导。
如果希望制作一行或更多行的无效变量表(作为注释行)，请使用编辑 > 无效行菜单命令或工具栏中相应的符号。

20.4.7 实例

20.4.7.1 在变量表中输入地址的实例

允许的地址：	数据类型：	实例(英语助记符)：
输入 | 输出 | 位存储器	BOOL	I 1.0 | Q 1.7 | M 10.1
输入 | 输出 | 位存储器	BYTE	IB 1 | QB 10 | MB 100
输入 | 输出 | 位存储器	WORD	IW 1 | QW 10 | MW 100
输入 | 输出 | 位存储器	DWORD	ID 1 | QD 10 | MD 100
I/O (输入 | 输出)	BYTE	PIB 0 | PQB 1
I/O (输入 | 输出)	WORD	PIW 0 | PQW 1
I/O (输入 | 输出)	DWORD	PID 0 | PQD 1
定时器	TIMER	T 1
计数器	COUNTER	C 1
数据块	BOOL	DB1.DBX 1.0
数据块	BYTE	DB1.DBB 1
数据块	WORD	DB1.DBW 1
数据块	DWORD	DB1.DBD 1

注意

不允许使用条目“DB0...”，因为它已在内部使用。

在强制值窗口中

- 在强制S7-300模块时，只允许输入、输出和I/O (输出)。
- 在强制S7-400模块时，只允许输入、输出、位存储器和I/O (输入/输出)。

20.4.7.2 输入连续地址范围的实例

打开变量表，使用菜单命令**插入 > 变量范围**，调用“变量的插入范围”对话框。

在对话框条目中，下列用于位存储器的行被插入变量表：

- 起始地址：M 3.0
- 编号： 10
- 显示格式：BIN

地址	显示格式
M 3.0	BIN
M 3.1	BIN
M 3.2	BIN
M 3.3	BIN
M 3.4	BIN
M 3.5	BIN
M 3.6	BIN
M 3.7	BIN
M 4.0	BIN
M 4.1	BIN

注意，在此例中，“地址”列中的标识在第八个条目后改变。

20.4.7.3 输入修改和强制值的实例

位地址

可能的位地址	允许的修改/强制值
I1.0	true
M1.7	false
Q10.7	0
DB1.DBX1.1	1
I1.1	2#0
M1.6	2#1

字节地址

可能的字节地址	允许的修改/强制值
IB 1	2#00110011
MB 12	b#16#1F
MB 14	1F
QB 10	'a'
DB1.DBB 1	10
PQB 2	-12

字地址

可能的字地址	允许的修改/强制值
IW 1	2#0011001100110011
MW12	w#16#ABCD
MW14	ABCD
QW 10	b#(12,34)
DB1.DBW 1	'ab'
PQW 2	-12345
MW3	12345
MW5	s5t#12s340ms
MW7	0.3s或0.3s
MW9	c#123
MW11	d#1990-12-31

双字地址

可能的双字地址	允许的修改/强制值
ID 1	2#00110011001100110011001100110011
MD 0	23e4
MD 4	2
QD 10	dw#16#abcdef10
QD 12	ABCDEF10
DB1.DBD 1	b#(12,34,56,78)
PQD 2	'abcd'
MD 8	l# -12
MD 12	l#12
MD 16	-123456789
MD 20	123456789
MD 24	t#12s345ms
MD 28	tod#1:2:34.567
MD 32	p#e0.0

定时器

可能的“定时器”类型的地址	允许的修改/强制值	解释
T 1	0	转换到毫秒(ms)
T 12	20	转换到毫秒
T 14	12345	转换到毫秒
T 16	s5t#12s340ms	
T 18	3	转换到1秒300毫秒
T 20	3s	转换到1秒300毫秒

修改定时器只影响值，不影响状态。这意味着可以将定时器T1的值修改为0，无需改变A T1逻辑操作的结果。

字符串5t、s5time可以用大写或小写写入。

计数器

可能的“计数器”类型的地址	允许的修改/强制值
C 1	0
C 14	20
C 16	c#123

修改计数器只影响值，不影响状态。这意味着可以将计数器C1的值修改为0，无需改变A C1逻辑操作的结果。

20.5 建立到CPU的连接

20.5.1 建立到CPU的连接

为了能监视或修改在当前变量表(VAT)中输入的变量，必须建立到适当的CPU的连接。可以将每个变量表与不同的CPU进行链接。

显示在线连接

如果存在在线连接，那么变量表窗口标题列中的术语“ONLINE”指示该情况。根据CPU，状态栏中会显示“RUN”、“STOP”、“DISCONNECTED”或“CONNECTED”工作状态。

建立到CPU的在线连接

如果不存在到所要求的CPU的在线连接，可使用菜单命令**PLC > 连接到 > ...**，定义一个到所要求的CPU的连接，从而可以监视或修改变量。

中断到CPU的在线连接

使用菜单命令**PLC > 断开**，可以中断变量表和CPU之间的连接。

注意

如果通过菜单命令**表格 > 新建**来创建一个未命名的变量表，那么在定义了该变量表时，可以将其连接到最近组态的已组态CPU上。

20.6 监视变量

20.6.1 监视变量简介

可使用下列方法来监视变量：

- 通过菜单命令**变量 > 监视器**激活监视功能。在变量表中根据所设置的触发点和触发频率显示选中变量的值。如果将触发频率设置为”每个周期”，那么可以通过菜单命令**变量 > 监视器**取消选择监视功能。
- 使用菜单命令**变量 > 更新监视值**，可随即更新选中变量的值。在变量表中显示选中变量的当前值。

通过ESC放弃“监视”

如果在“监视”功能激活时，按下ESC，那么不经过查询就会终止该功能。

20.6.2 定义用于监视变量的触发器

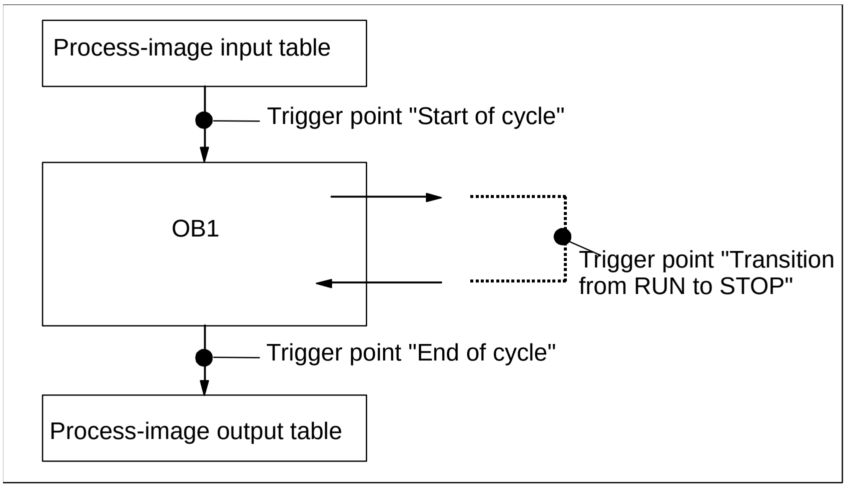
可以在程序处理期间的特定点(触发点)处，在编程设备上显示用户程序内单个变量的当前值，以进行监视。
选择了触发点，就确定了显示变量监视值的时间点。

可使用菜单命令**变量 > 触发器**来设置触发点和触发频率。

触发器	可能的设置
触发点	周期开始 周期结束 从RUN转换到STOP
触发频率	一次 每个周期

触发点

下图显示了触发点的位置。



为了在“状态值”列中显示修改值，应该将用于监视的触发点设置为“周期开始”，将用于修改的触发点设置为“周期结束”。

立即触发

可使用菜单命令**变量 > 更新监视值**来更新选中变量的值。采用该命令表示“立即触发”并尽快执行，不参考用户程序中的任何一点。这些功能主要用于在STOP模式中进行监视和修改。

触发频率

下表显示了触发频率对监视变量的影响：

	触发频率：一次	触发频率：每个周期
监视变量	*更新一次* 取决于触发点	*通过所定义的触发器监视* 测试块时，可精确跟踪处理进程。

20.7 修改变量

20.7.1 关于对变量进行修改的说明

修改变量时，可采用下列方法：

- 使用菜单命令**变量 > 修改**，激活修改功能。根据设置的触发点和触发频率，用户程序为变量表中选择的变量应用修改值。如果设置的触发频率为“每周期”，可以使用菜单命令**变量 > 修改**来关闭修改功能。
- 可以更新一次所选变量的值，然后立即使用菜单命令**变量 > 激活修改值**。

强制和启用外围设备输出(PQ)功能提供了其它的可能性。

修改变量时的注意事项：

- 只有在开始修改时，那些在变量表中可见的地址才被修改。
  如果您在开始修改后减少变量表的可见区域大小，可能导致能被修改的地址不可见。
  如果扩大变量表的可视区域，可能会有一些可见的地址没有被修改。
- 修改无法撤消(例如用**编辑 > 撤消**)。

危险

过程正在运行时改变变量值，如果此时功能或程序发生错误，可能会导致财产或人员的严重损失。
在执行“修改”功能以前，确保不会发生危险的状况。

使用ESC键中止“修改”

当“修改”功能正在进行时，按下ESC键，将不作任何询问中止功能。

20.7.2 定义用于修改变量的触发器

可在程序处理期间，在指定点处给用户程序的单个变量分配固定值(一次或每个周期)。

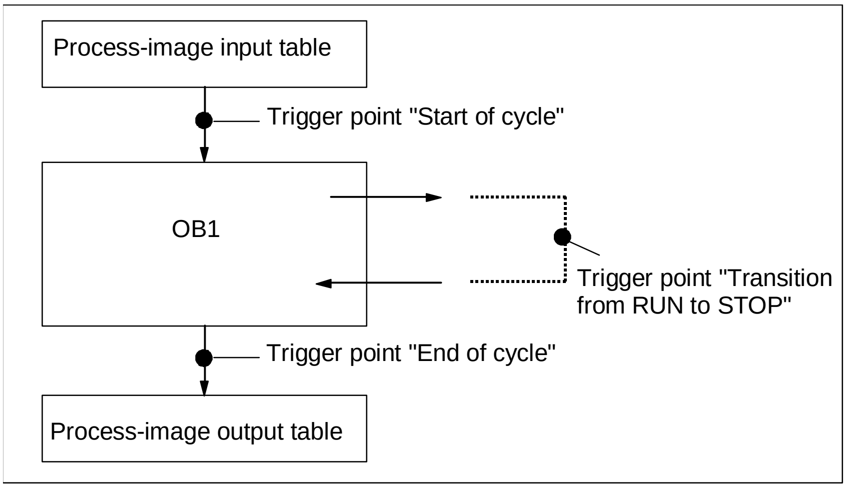
选择了触发点，就确定了将修改值分配给变量的时间点。

可使用菜单命令**变量 > 触发器**来设置触发点和触发频率。

触发器	可能的设置
触发点	周期开始 周期结束 从RUN转换到STOP
触发频率	一次 每个周期

触发点

下图显示了触发点的位置。



触发点的位置显示：

- 修改输入仅对“周期开始”触发点有用(相当于用户程序 OB1 开始)，因为否则在修改后会更新输入的过程映像，从而重写)。
- 修改输出仅对“周期结束”触发点有用(相当于用户程序 OB1 结束)，因为否则用户程序重写输出的过程映像)。

为了在“状态值”列中显示修改值，应该将用于监视的触发点设置为“周期开始”，将用于修改的触发点设置为“周期结束”。

修改变量时，下列各项适用于触发点：

- 如果将触发频率设为“一次”，那么当不能修改选中的变量时，出现一个消息。
- 触发频率为“每个周期”时，不出现消息。

立即触发

可使用菜单命令**变量 > 激活修改值**来修改选中变量的值。采用该命令表示“立即触发”并尽快执行，不参考用户程序中的任何一点。该功能主要用于在STOP模式中进行修改。

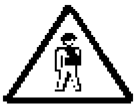
触发频率

下表显示了所设置的触发条件对修改变量的影响：

	触发频率：一次	触发频率：每个周期
修改变量	*激活一次* 可以给变量赋值一次，与触发点无关。	*通过预定义的触发器进行修改* 通过分配固定值，可以为用户程序模拟特定条件，并以此来调试已经编程的功能。

20.8 强制变量

20.8.1 在强制变量时的安全措施



当心造成人员伤亡和财产损坏

注意，当使用“强制”功能时，任何不正确的操作都可能会：

- 造成人员伤亡，或
- 造成机器或整个工厂的损坏



警告

- 在启动强制功能前，应该查明没有人同时在同一CPU上执行此功能。
- 强制作业只能用菜单命令**变量 > 停止强制**来删除或终止。关闭强制值窗口或退出“监视和修改变量”应用程序不会删除强制作业。
- 强制不能撤销(例如用**编辑 > 撤销**)。
- 要阅读关于强制和修改变量之间的区别的信息。
- 如果CPU不支持强制功能，与强制动作链接的变量菜单中的所有菜单命令都是取消激活的。

如果使用菜单命令**变量 > 允许外围设备输出**来取消激活输出禁用，所有的强制输出模块都会输出它们的强制值。

20.8.2 关于对变量进行强制的说明

可以为用户程序的单个变量分配固定值，这样，即使是CPU中正在执行的用户程序，也不能对其加以修改或覆盖。对此的要求是CPU必须支持该功能(例如，S7-400 CPU)。通过为变量分配固定的值，可以为用户程序设置特定的状况，然后以此来测试编写的功能。

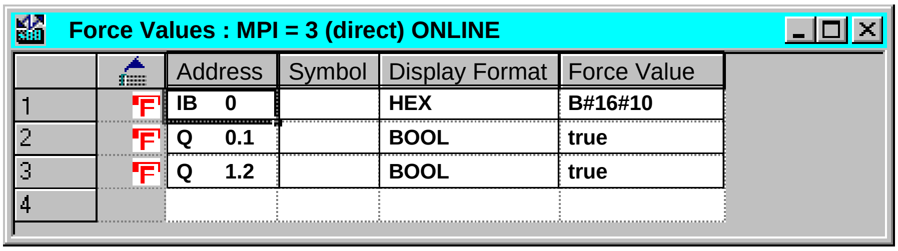
“强制值”窗口

只有当“强制值”窗口是激活的，才可以选择强制菜单命令。

要显示该窗口，可选择菜单命令**变量 > 显示强制值**。

对于每个CPU，只应当打开一个“强制值”窗口。在该窗口中显示激活的强制作业的变量，以及它们各自的强制值。

强制值窗口的实例



当前在线连接的名称显示在**标题栏**中。

从CPU中读取的强制作业的数据和时间显示在**状态栏**中。

如果没有激活的强制作业，则窗口为空。

在“强制值”窗口中，**显示变量**的不同方法具有以下含义：

显示	含义
粗体：	已经在CPU中分配了固定值的变量。
常规：	正在编辑的变量。
呈灰色：	机架中不存在/未插入的模块的变量 或 地址错误的变量；显示出错消息。

使用变量表中的强制地址

如果要将变量表中的变量输入到强制值窗口，选择表格和所需的变量。下一步，调用菜单命令**变量 > 强制值**，打开强制值窗口。模块可以强制的变量将被输入到强制值窗口中。

使用CPU的强制作业或建立一个新的强制作业

如果“强制值”窗口已经打开并已激活，将会显示另外的消息：

- 如果确认它，窗口中的改变将以CPU上存在的强制作业来覆盖。可以使用菜单命令**编辑 > 撤消**来恢复先前窗口的内容。
- 如果取消它，则将保持窗口的当前的内容。
  要将“强制值”窗口的内容保存为变量表，可以使用菜单命令**表格 > 另存为**或选择菜单命令**变量 > 强制**：它将把窗口的当前内容作为新的强制作业写入CPU
  。

只能在变量表中监视和修改变量，而不能在“强制表”窗口中进行。

删除强制值

调用菜单命令**变量 > 显示强制值**来打开强制值窗口。然后，调用菜单命令**变量 > 删除强制**从所选的CPU中删除强制值。

保存强制值窗口

可以将强制值窗口中内容保存到变量表中。使用菜单命令**插入 > 变量表**，可以将保存的内容重新插入到强制值窗口中。

关于强制值窗口中符号的注意事项

除非是从不包含符号的其它应用程序中打开“监视和修改变量”应用程序，否则将输入上次激活的窗口中的符号。

如果不能输入符号名称，将隐藏“符号”列。在这种情况下，将取消激活菜单命令**选项 > 符号表**。

20.8.3 强制变量和修改变量之间的差别

下表总结了强制变量和修改变量之间的差别：

特性/功能	S7-400中的强制 (包括 CPU 318-2DP)	S7-300中的强制 (不包括 CPU 318-2DP)	修改
位存储器(M)	是	–	是
定时器和计数器(T、C)	–	–	是
数据块(DB)	–	–	是
外围设备的输入(PIB、PIW、PID)	是	–	–
外围设备的输出(PQB、PQW、PQD)	是	–	是
输入与输出(I、Q)	是	是	是
用户程序可以覆盖修改/强制值	–	是	是
不必中断有效地替换强制值	是	是	–
当应用程序退出后，变量保持它们的值	是	是	–
当与CPU的连接中断后，变量仍保持它们的值	是	是	–
允许的寻址错误： 例如 IW1 修改/强制值：1 IW1 修改/强制值： 0	–	–	最后一个变为有效值
设置触发器	总是立即触发	总是立即触发	一次或每个周期
功能仅在激活窗口的可视区域影响变量	影响所有强制值	影响所有强制值	是

注意

- 通过“启用外围设备的输出”，使强制外围设备输出的强制值在相应的输出模块上生效；然而，外围设备输出的修改值则不会。
- 对于强制变量，变量始终具有强制值。每次读取用户程序时，都会读取该值。所有形式的写访问都无效。
- 对于永久的修改，程序的读取访问有效并一直持续到下一个触发点。

---

21 使用编程状态测试

您可以通过显示每条指令的程序状态(RLO、状态位)或相应寄存器的内容来测试程序。在“自定义”对话框的“LAD/FBD”标签里，可以定义显示信息的范围。在“LAD/STL/FBD：程序块”窗口中使用菜单命令**选项 > 自定义**。

警告

过程正在运行时测试程序，如果此时功能或程序发生错误，可能会导致财产或人员的严重损失。

执行该功能前确保不会发生危险的情况。

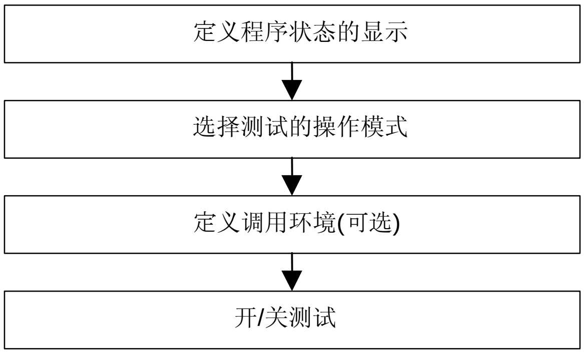
要求

要显示程序状态，需要满足下列要求：

- 必须保存没有错误的块，然后再将它下载到CPU中。
- CPU和用户程序必须正在运行。

监视程序状态的基本步骤

强烈建议您不要调用整个程序来调试，而是逐个的调用块，然后单独对其进行调试。应当从调用体系最外的嵌套层的块开始，例如，在OB1中调用它们，然后通过监视和修改变量，为块创建要测试的环境。



要设置断点，以单步模式来执行程序，必须设置测试操作模式(参见菜单命令**调试 > 操作**)。这些测试功能不能用于过程操作模式。

21.1 程序状态显示

程序状态的显示是循环更新的。它从选择的程序段开始。

在LAD和FBD中预设颜色

- 状态实现：绿色实线
- 状态没有实现：蓝色点划线
- 状态未知：黑色实线

线线类型和颜色的预设值可以在菜单命令**选项 > 自定义**、“LAD/FBD”标签页下改变。

元素状态

- 触点的状态是：
  - 如果地址具有值“1”代表实现
  - 如果地址具有值“0”代表没有实现
  - 如果地址值为未知则代表未知
- 具有启用输出(ENO)的元素的状态对应于将ENO输出值作为地址触点的状态。
- 具有Q输出的元素的状态对应于具有地址值触点的状态。
- 如果调用后BR位被置位，那么CALL状态将实现。
- 如果执行跳转(即如果跳转条件满足)，那么跳转指令的状态将实现。
- 如果启用的输出没有连接，具有启用输出(ENO)的元素以黑色显示。

线状态

- 如果未穿过线或如果线状态未知，则这些线是黑色的。
- 从母线开始的线状态始终(为“1”)。
- 从并联分支开始的线状态始终(为“1”)。
- 如果元素前的线状态和元素状态均为“1”，那么元素后的线状态也将为“1”。
- 如果在NOT前的线状态不为“1”，那么在NOT后的线状态为“1”(反之亦然)。
- 如果满足以下条件，则线状态将在一些线相交**后**为“1”：
  - 在相交**前**至少有一条线的状态为“1”。
  - 在分支前的线状态为“1”。

参数状态

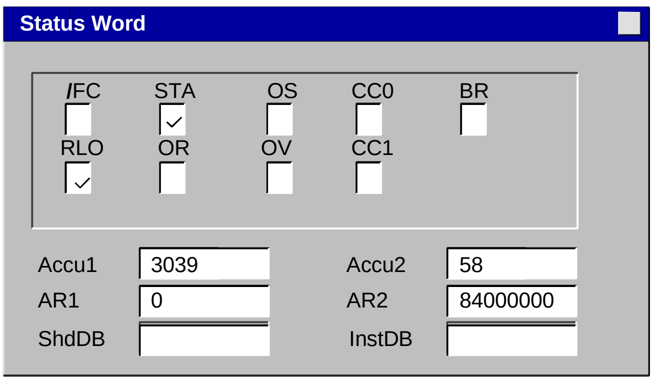
- 以**粗体**显示的参数值为当前值。
- 以细体字显示的参数值来自前一个周期；当前扫描周期不处理程序部分。

21.2 单步模式/断点测试须知

当在单步模式下测试时，可以进行下列工作：

- 逐个语句地执行程序(单步)
- 设置断点

不是所有的可编程控制器都可以使用“以单步模式测试”(参见相关可编程控制器的文档)。



要求

- 必须设置测试操作模式。单步模式测试不能在操作模式下进行(参见菜单命令**调试 > 操作**)。
- 在单步模式下测试只可能在语句表中进行。对于梯形图或功能块图中的块，必须使用菜单命令**视图 > STL**来改变视图。
- 不得保护块。
- 必须在线打开块
- 不得在编辑器中改变打开的块。

断点的数目

断点的数目为变量，由下列事项决定：

- 已设置的断点数目
- 正在运行的变量状态数目
- 正在运行的程序状态数目

参考可编程控制器文档，可查明它是否支持单步模式测试。

在“调试”菜单中可以找到能用来设置、激活或删除断点的菜单命令。也可以使用断点工具栏中的图标选择那些菜单命令。使用菜单命令**视图 > 断点条**来显示断点工具栏。

允许的测试功能

- 监视/修改变量
- 模块信息
- 工作模式

危险

设备状态在HOLD模式下有出现危险的风险。

21.3 HOLD模式须知

如果程序遇到断点，可编程控制器转到HOLD工作模式。

在HOLD模式中的LED显示

- LED RUN 闪烁
- LED STOP 亮

在HOLD模式中的程序处理

- 在HOLD模式中，不处理S7代码，这意味着不再进行优先级处理。
- 冻结所有的定时器：
  - 不处理定时器单元
  - 暂停所有监视时间
  - 暂停时间控制水平的基本时钟率
- 实时时钟继续运行
- 出于安全原因，在HOLD模式中始终禁止输出(“输出禁止”)。

在HOLD模式下出现电源故障后的状态

- 在HOLD模式下发生电源故障及随后恢复上电后，具有后备电池的可编程控制器切换到STOP模式并保持该模式。CPU不执行自动重新启动(热重启)。在STOP模式下，可以确定如何继续处理(例如，通过设置/复位断点，执行手动重新启动)。
- 没有后备电池的可编程控制器没有“掉电保护”，因此，无论以前的工作模式如何，当电源重新上电时都会执行自动热重启。

21.4 数据块的程序状态

从STEP 7版本5开始，可以在数据视图中查看在线数据块。可以通过在线数据块或离线数据块激活显示。在这两种情况下，会显示可编程控制器中的在线数据块的内容。

在程序状态启动前，不得修改数据块。如果在线数据块和离线数据块的结构有所不同(声明)，可以根据要求直接将离线数据块下载到可编程控制器中。

数据块必须位于“数据视图”中，以便在线值可以在“真实值”栏中显示。只能更新画面中可见的数据块部分。当状态激活时，不能切换到声明视图。

在进行更新时，在状态栏中可以看见绿色条，并显示工作模式。

数值以相应数据类型的格式显示；格式不能改变。

在程序状态结束后，“真实值”栏再次显示在程序状态之前有效的内容。不能将更新后的在线值传送到离线数据块中。

更新数据类型：

所有基本数据类型都在共享数据块中更新，也在所有的背景数据块的声明(输入/输出/输入-输出/静态)中更新。

某些数据类型不能更新。当程序状态激活时，包含没有更新数据的“真实值”栏中的域用灰色背景显示。

- 不更新复杂的数据类型DATE_AND_TIME和字符串。
- 在复杂数据类型ARRAY、STRUCT、UDT、FB和SFB中，只更新那些基本数据类型的元素。
- 在背景数据块的INOUT声明中，只显示复杂数据类型的指针，而不显示数据类型元素本身。不更新指针。
- 不更新参数类型

21.5 为程序状态设置显示

可以在语句表、功能块图或梯形图中设置程序状态的显示。

要设置显示，操作如下：

1. 选择菜单命令**选项 > 自定义**。
2. 在“自定义”对话框中，选择“STL”标签页或“LAD/FBD”标签页。
3. 选择所要求的用于测试程序的选项。可以显示下列状态域。

激活...	显示...
状态位	状态位;状态字的第2位
RLO	状态字的第1位 显示逻辑操作或算术比较的结果
标准状态	累加器1的内容
地址寄存器1/2	使用寄存器间接寻址的相关地址寄存器的内容(区域内或跨区域)
Akku2	累加器2的内容
数据块寄存器1/2	第一个和/或第二个打开的数据块的数据块寄存器的内容
间接	间接存储器参考；指针参考(地址)，没有地址内容参考； 仅适用于存储器间接寻址，不能用于寄存器间接寻址。 若相应的指令出现在语句中时，定时器字或计数器字的内容
状态字	状态字的所有状态位

21.6 为测试设置模式

步骤

1. 使用菜单命令**调试 > 操作**，显示所设置的测试环境。
2. 选择所需要的操作模式。可以在测试操作和过程操作之间选择。

操作模式	解释
测试操作	所有的测试功能都可用，且不受限制。 CPU扫描周期时间会明显增加，这是因为，例如，程序回路中的语句状态在每个周期都被记录。
过程操作	测试功能程序状态被限定为要保证在扫描周期时间上可能出现的负载为最小。 • 这意味着，例如，不允许任何调用条件。 • 程序回路的状态显示在返回点处被放弃。 • 不能进行HOLD测试功能和单步执行程序。

注意

当分配CPU参数时，如果设置操作模式，只能通过改变参数来改变模式。否则，在显示的对话框中改变模式。

22 使用模拟程序进行测试(可选择的软件包)

22.1 使用模拟程序S7 PLCSIM (可选择的软件包)进行测试

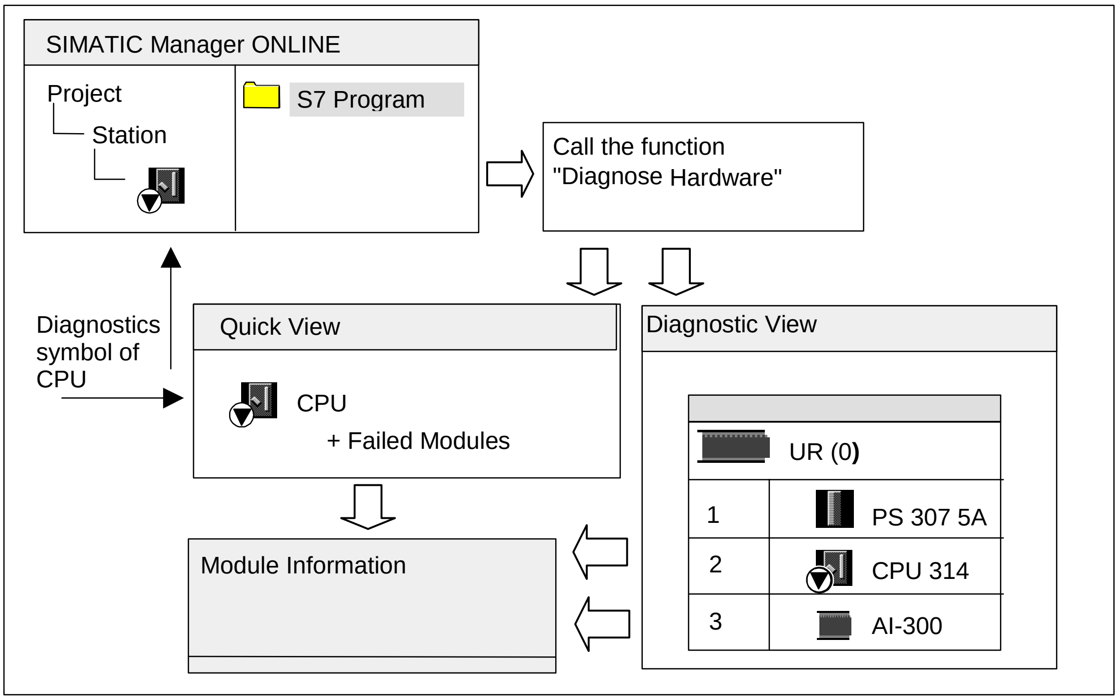
使用可选择的软件包PLC Simulation，可以在计算机或编程设备(如Power PG)中的模拟可编程控制器上运行和测试程序。因为模拟完全由STEP 7软件实现，所以不需要任何S7硬件(CPU或信号模块)。使用模拟S7 CPU，可以测试和维护S7-300和S7-400 CPU的程序。

此应用程序提供简单的用户界面，以用于监视和修改在程序中使用的各种参数(例如，用于开、关输入)。当程序由模拟CPU处理时，也可以在STEP 7软件中使用各种应用程序。例如，可以用变量表监视和修改变量。

23 诊断

23.1 诊断硬件和故障排除

通过出现的诊断符号，您可查看是否有可供模块使用的诊断消息。诊断符号说明了相应模块的状态，而且，对于CPU，也说明了其工作模式。
当调用功能“诊断硬件”后，诊断符号将会显示在在线视图以及快速视图(缺省设置)或诊断视图的项目窗口中。双击快速视图或诊断视图中的诊断符号，可启动“模块信息”应用程序来显示详细的诊断信息。



如何定位故障

1. 使用菜单命令**视图 > 在线**打开项目的在线窗口。
2. 打开所有的站，以便在其中组态的可编程模块均为可见。
3. 查看是哪个CPU正在显示诊断符号，其指示了错误或故障。按F1键打开解释诊断符号的帮助页面。
4. 选择要检查的站。
5. 选择菜单命令**PLC > 诊断/设置 > 模块信息**以显示该站中CPU的模块信息。
6. 选择菜单命令**PLC > 诊断/设置 > 诊断硬件**以显示该站中CPU和 故障 模块的“快速视图”。快速视图的显示已设置为缺省值(菜单命令**选项 > 自定义**，“视图”标签)。
7. 选择快速视图中的故障模块。
8. 点击“模块信息”按钮以获取关于该模块的信息。
9. 点击快速视图中的“在线打开站”按钮，以显示诊断视图。诊断视图包括了按照其插槽顺序排列的站中的 所有 模块。
10. 双击诊断视图中的模块，以便显示模块信息。采用该方式，您也可获得那些没有故障因而没有显示在快速视图中的模块的信息。

您当然不必执行所有的这些步骤；一旦您获得所需要的诊断信息，您就可停止。

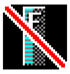
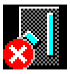
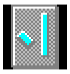
23.2 在线视图中的诊断符号

在在线项目窗口和具有组态表在线视图的硬件配置窗口中，显示诊断符号。

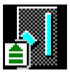
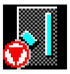
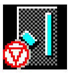
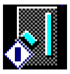
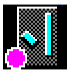
诊断符号便于检测故障。只需看一眼模块符号，就知道有没有诊断信息。如果没有出现故障，那么所显示的模块类型符号上不带附加的诊断符号。

如果模块有诊断信息，那么除显示模块符号外，还显示诊断符号，或以较低的对比度显示模块符号。

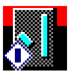
模块的诊断符号(实例：FM/CPU)

符号	含义
	预设定和实际组态之间不匹配：已组态的模块不存在或插入了一个不同类型的模块
	故障：模块出现故障。 可能的原因：诊断中断、I/O访问错误或检测到错误LED
	不能进行诊断：没有在线连接，或CPU没有将诊断信息返回模块(例如，电源或子模块)。

工作模式的诊断符号(实例：CPU)

符号	模式
	启动
	停止
	停止 在多值计算操作中，由另一个CPU的STOP模式触发
	运行
	保持

强制诊断符号

符号	模式
	该模块上的变量是强制的，即该模块的用户程序中的变量分配有固定值，这些值不能由程序修改。 强制符号还可与其它符号组合出现(在此，与运行模式的符号组合)。

更新诊断符号的显示

必须激活合适的窗口。

- 按下F5或
- 在窗口中选择菜单命令**视图 > 更新**。

23.3 诊断硬件快速视图

23.3.1 调用快速视图

快速视图提供一种使用“诊断硬件”的快捷方式，其中的信息量少于在HW Config的诊断视图中的详细显示信息。当调用“诊断硬件”功能时，快速视图作为缺省显示。

显示快速视图

可以在SIMATIC管理器中，使用菜单命令**PLC > 诊断/设置 > 诊断硬件**来调用该功能。

可按如下方式使用该菜单命令：

- 如果选择了一个模块或S7/M7程序，那么在项目的在线窗口中。
- 如果在“可访问节点”窗口中选择一个节点(“MPI=...”)，那么该条目属于CPU。

从所显示的组态表中，可以选择希望显示其模块信息的模块。

23.3.2 快速视图中的信息功能

快速视图中会显示如下信息：

- 在线连接到CPU的数据
- CPU的诊断符号
- 被CPU检测出故障的模块的诊断符号(例如，诊断中断、I/O访问错误)
- 模块类型和模块地址(机架、插槽、具有站编号的DP主站系统)。

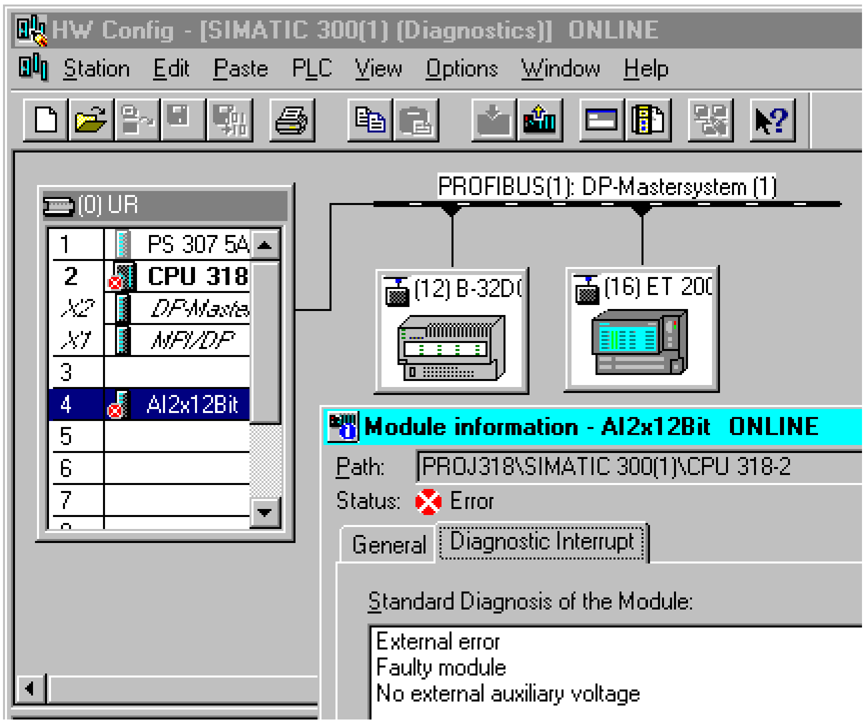
快速视图中的其它诊断选项

- 显示模块信息
  通过点击“模块信息”按钮，可以调用该对话框。该对话框根据所选模块的诊断能力，显示详细的诊断信息。特别是，通过CPU的诊断信息，可以显示缓冲诊断区中的条目。
- 显示诊断视图
  使用“在线打开站”按钮显示诊断视图，可以打开一个对话框，该对话框与快速视图不同，包含整个站的图形总览以及组态信息。
  它侧重于在“CPU/故障模块”列表中高亮显示的模块。

23.4 诊断硬件诊断视图

23.4.1 调用诊断视图

使用本方法，可以为机架中的所有模块打开“模块信息”对话框。诊断视图(组态表)显示机架级的站以及具有各自模块的DP站的实际结构。



注意

- 如果已离线打开组态表，则也可使用菜单命令**站 > 在线打开**来获得组态表的在线视图。
- 根据模块的诊断能力，在“模块信息”对话框中显示不同数目的标签。
- 在“可访问节点”窗口中，只有具有本身节点地址(MPI或PROFIBUS地址)的模块才可见。

在SIMATIC管理器中从项目的在线视图中调用

1. 在SIMATIC管理器的项目视图中，使用菜单命令**视图 > 在线**，建立到可编程控制器的在线连接。
2. 选择一个站，并双击打开该站。
3. 接着打开其中的“硬件”对象。打开诊断视图。

现在可以选择一个模块，然后使用菜单命令**PLC > 诊断/设置 > 模块信息**调用其模块信息。

在SIMATIC管理器中从项目的离线视图中调用

执行下列步骤：

1. 从SIMATIC管理器的项目视图中选择一个站，然后双击打开该站。
2. 接着打开其中的“硬件”对象。打开组态表。
3. 选择**站 > 在线打开**菜单命令。
4. 打开HW Config的诊断视图，同时打开由模块(例如，CPU)确定的站组态。模块状态用符号表示。请参见在线帮助，获取各种符号的含义信息。在一个单独的对话框中列出故障模块和丢失的已组态模块。从该对话框中，可以直接导航到其中一个选中的模块(“跳转到”按钮)。
5. 双击对其状态感兴趣的模块符号。该模块状态的详细分析在一个具有标签(根据模块类型)的对话框中给出。

从SIMATIC管理器的“可访问节点”窗口中调用

执行下列步骤：

1. 在SIMATIC管理器中使用菜单命令**PLC > 显示可访问节点**，打开“可访问节点”窗口。
2. 在“可访问节点”窗口中选择一个节点。
3. 选择菜单命令PLC > 诊断/设置 > 诊断硬件。

注意

在“可访问节点”窗口中，只有具有本身节点地址(MPI或PROFIBUS地址)的模块才可见。

23.4.2 诊断视图中的信息功能

与快速视图相比，诊断视图显示在线可用的整个站组态。这包含：

- 机架配置
- 所有已组态模块的诊断符号
  从这些诊断符号中，可以读取每个模块的状态，如果是CPU模块，则可读取工作模式。
- 组态的模块类型、订货号、地址详细资料以及注释。

诊断视图中的附加诊断选项

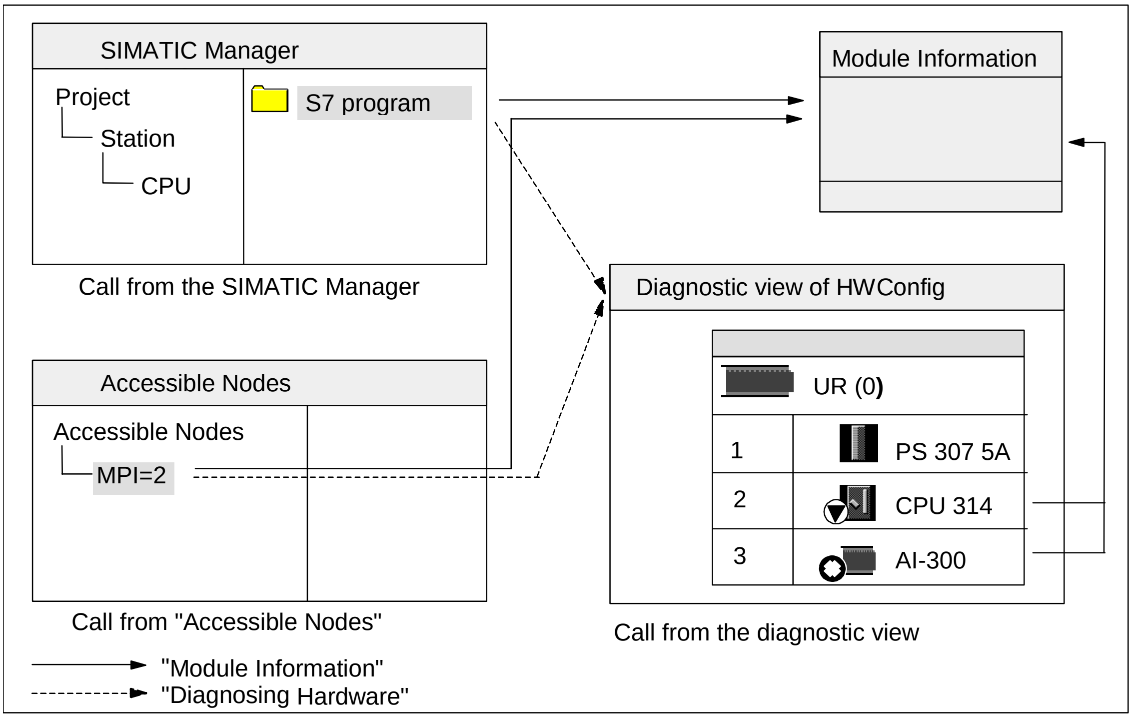
通过双击模块，可以显示该模块的工作模式。

23.5 模块信息

23.5.1 用于显示模块信息的选项

可从不同的起点显示“模块信息”对话框。下列步骤是调用模块信息常使用的方法实例：

- 在SIMATIC管理器中，通过“在线”或“离线”项目视图窗口。
- 在SIMATIC管理器中，通过“可访问节点”窗口
- 在HW Config的诊断视图中



为了显示**具有本身节点地址的模块**的状态，可要求在线连接到可编程控制器。可通过项目的在线视图或通过“可访问节点”窗口建立该连接。

23.5.2 模块信息功能

可在“模块信息”对话框内的各种标签中查找每个模块的信息功能。在激活状态下显示时，只显示与选中模块有关的那些标签。

功能/标签	信息	使用方法
常规	所选择的模块上的标识数据；例如，订货号、版本号、状态、机架中的插槽	将来自插入模块的在线信息与已组态模块的数据进行比较
诊断缓冲区	诊断缓冲区中的事件总览以及选中事件的详细信息	查找引起CPU进入STOP模式的原因，并在选中的模块上评估导致该原因的事件 通过诊断缓冲区，可以在以后分析系统中的错误，查找引起STOP的原因或追踪并归类单个诊断事件的发生。
诊断中断	选中模块的诊断数据	评估模块故障的原因
DP从站诊断	选中DP从站的诊断数据(符合EN 50170)	在DP从站中评估故障原因

功能/标签	信息	使用方法
存储器	存储器容量。选中CPU或M7功能块的工作存储器、装入存储器和保持性存储器的当前用途	在将新模块或扩展模块传送到CPU之前，检查CPU/功能模块中是否有足够可用的装入存储器，否则就压缩存储器内容。
扫描周期	选中CPU或M7功能模块的最长、最短以及最后一次扫描周期的持续时间	持续检查已组态的最小周期、最大周期和当前周期
时间系统	当前时间、工作小时和 同步时钟的信息(同步时间间隔)	显示和设置模块的时间与日期，并检查时间同步
性能数据	选中模块(CPU/FM)的地址区和可用的块	在创建用户程序之前以及期间，检查CPU是否满足执行用户程序的要求；例如，装入存储器大小或过程映像大小
块 (可从“性能数据”标签中打开)	显示选定模块供应范围内的所有可用的块类型，列出可用于该模块的OB、SFB和SFC	检查用户程序可包含或调用哪些可在选中CPU上运行的标准块。
通讯	传输率、通讯连接总览、通讯负载以及选中模块通讯总线上的最大消息帧的大小	确定可使用多少个、哪个CPU或M7 F7连接，以及正在使用的数目
栈	**堆栈**标签：只能在STOP模式或HOLD模式中调用。 显示用于选中模块的B栈。然后还可显示I栈、L栈以及嵌套栈，并跳转到中断块中的错误位置。	确定转换到STOP模式的原因并更正块

所显示的附加信息

每个标签有下列显示信息：

- 到选中模块的在线路经
- 相应CPU的工作模式(例如，RUN、STOP)
- 选中模块的状态(例如，错误、正常)
- 如果具有各自的工作模式(例如，CP 342-5)，那么显示选中模块的工作模式(例如，RUN、STOP)

如果从“可访问节点”窗口中打开了非CPU模块的模块信息，就不能显示CPU本身的工作模式以及选中模块的状态。

同时显示大量模块

可同时显示大量模块的模块信息。为此，必须转到每个模块环境，选择另一个模块，然后调用该模块的模块信息。然后显示另一个“模块信息”对话框。每个模块只能打开一个对话框。

更新模块信息显示

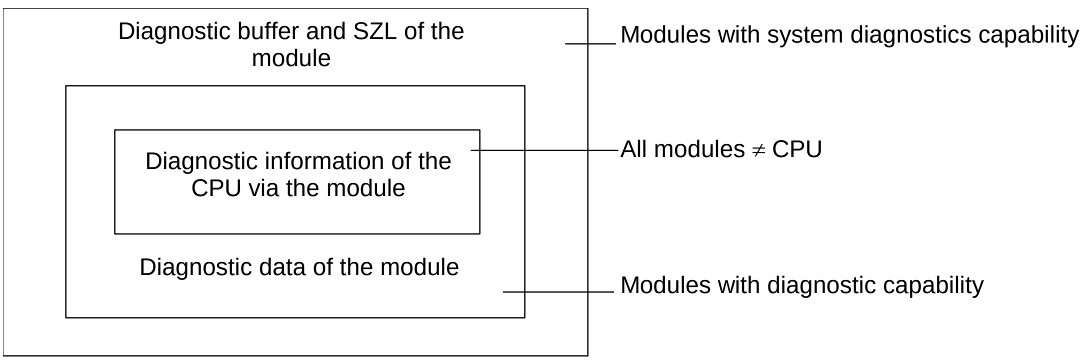
每次切换到“模块信息”对话框中的一个标签时，会重新从模块中读取数据。不过，显示一个页面时，不更新其内容。如果点击“更新”按钮，那么可以在不改变标签的情况下，从模块中读取数据。

23.5.3 与模块类型有关的信息范围

可以评估和显示的信息范围取决于：

- 所选的模块以及
- 调用模块信息的视图
  从组态表的在线视图或项目窗口中调用时，可以得到全范围的信息。
  从“可访问节点”窗口调用时，只能得到有限范围的信息。

根据信息范围，模块划分为“具有系统诊断能力”、“具有诊断能力”或“无诊断能力”类别。下图显示这些类别：



- 具有系统诊断能力的模块，例如，模块FM 351和FM 354
- 具有诊断能力的模块，大部分是模拟信号模块。
- 无诊断能力的模块，大部分为数字信号模块。

显示的标签

该表显示了在每种模块类型的“模块信息”对话框中出现的属性标签。

标签	**CPU或 M7 FM**	具有系统诊断能力的模块	具有诊断能力的模块	无诊断能力的模块	**DP**从站
常规	是	是	是	是	是
诊断缓冲区	是	是	–	–	–
诊断中断	–	是	是	–	是
存储器	是	–	–	–	–
扫描周期	是	–	–	–	–
时间系统	是	–	–	–	–
性能数据	是	–	–	–	–
栈	是	–	–	–	–
通讯	是	–	–	–	–
DP从站诊断	–	–	–	–	是
H状态[1)]	是	–	–	–	–
[1)]只用于H系统中的CPU					

除了标签属性页上的信息之外，具有工作模式的模块还显示工作模式。从组态表在线打开该对话框时，从CPU角度来显示模块的状态(例如，正常、故障、模块不可用)。

23.5.4 显示进行Y-链接之后的PA现场设备和DP从站的模块状态

从STEP 7 V5.1 Service Pack 3起，您可在进行DP/PA链接(IM 157)“之后”计算DP从站和PA现场设备的模块状态。

这将影响下列组态：

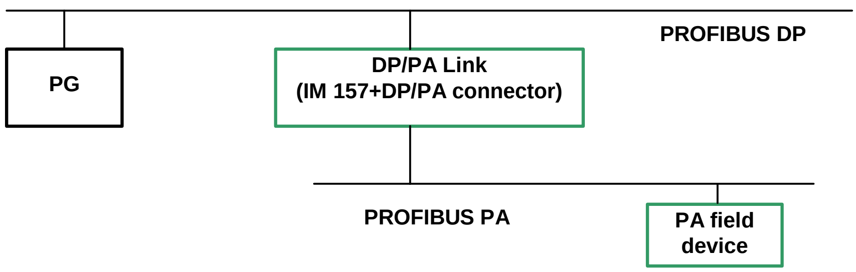
- 具有DP/PA连接器的IM 157，用于连接PROFIBUS-PA
- 作为冗余模块化接口模块的IM 157，用于连接非冗余PROFIBUS-DP (“Y-链接”)

在该组态中，编程设备(PG)将被连接到与DP/PA链接相同的PROFIBUS子网上。

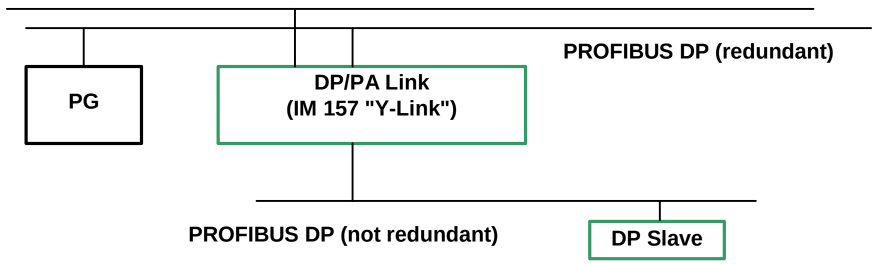
此外，存在另一个组态选项，在该选项中，PG将连接到工业以太网并将S7-400站发送到PROFIBUS子网上。

该设置的先决条件如下图所示：

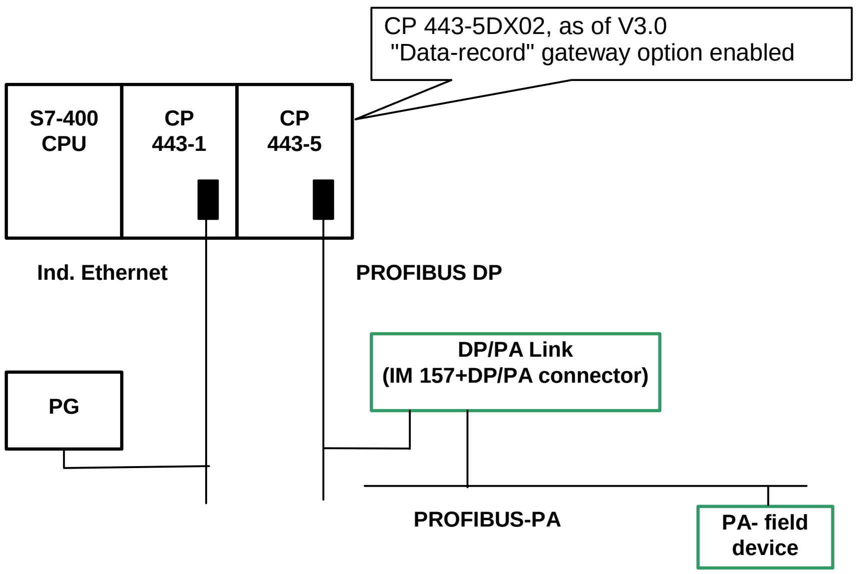
具有**DP/PA**连接器的**IM 157**，用于连接**PROFIBUS-PA**



作为**Y-**链接的**IM 157**



工业以太网中的**PG**



23.6 在STOP (停止)模式中进行诊断

23.6.1 确定停止原因的基本步骤

要确定CPU为何进入“停止”模式，可如下操作：

1. 选择已进入停止模式的CPU。
2. 选择菜单命令PLC > 诊断/设置 > 模块信息。
3. 选择“诊断缓冲区”标签。
4. 可以从诊断缓冲区的最后一个条目确定停止原因。

如果发生编程错误：

1. 例如，条目“由于没有加载编程错误OB而停止”表示CPU检测到一个程序错误，然后尝试启动(不存在的) OB来处理该编程错误。前一个条目指代实际的编程错误。
2. 选择与编程错误有关的消息。
3. 点击“打开块”按钮。
4. 选择“栈”标签。

23.6.2 停止模式中的栈内容

通过评估诊断缓冲区和栈内容，可以确定用户程序处理期间发生的故障的原因。

例如，如果由于编程错误或“停止”命令导致CPU进入停止模式，那么模块信息中的“栈”标签显示块栈。可以使用“I栈”、“L栈”和“嵌套栈”按钮来显示其它栈的内容。栈内容给出哪个块中的哪条指令导致CPU进入停止模式的信息。

B栈内容

B栈，或称块栈，列出了变为停止模式之前调用的所有块以及没有完全处理的块。

I栈内容

点击“I栈”按钮时，显示中断位置处的数据。I栈，或称中断栈，包含中断时有效的数据或状态，例如：

- 累加器内容和寄存器内容
- 打开的数据块及其大小
- 状态字的内容
- 优先级(嵌套等级)
- 中断块
- 中断后，继续进行程序处理的块

L栈内容

对于B栈中列出的每个块，通过选择该块并点击“L栈”按钮，可以显示相应的局部数据。

L栈，或称局部数据栈，包含发生中断时用户程序正在处理的块的局部数据值。

解释和评估所显示的局部数据要求非常熟悉系统。所显示数据的第一部分对应于块的临时变量。

嵌套栈内容

点击“嵌套栈”按钮时，显示中断位置处嵌套栈的内容。

嵌套栈是逻辑操作A(、AN(、O(、ON(、X(和XN(使用的存储区。

只有在中断时仍然打开括号表达式时，才激活该按钮。

23.7 检查扫描周期，避免出现时间错误

23.7.1 检查扫描周期，避免出现时间错误

模块信息中的“扫描周期”标签给出关于用户程序扫描周期的信息。

如果最长周期的持续时间接近所组态的最大扫描周期，则有因周期波动引发时间错误的危险。延长用户程序的最大周期(监视狗时间)可避免出现这种情况。

如果周期长度小于所组态的最小扫描时间，则由CPU/FM自动将该周期延长至所组态的最小周期。如果为CPU，则在该延长时间内处理背景OB (OB90) (如果已经下载)。

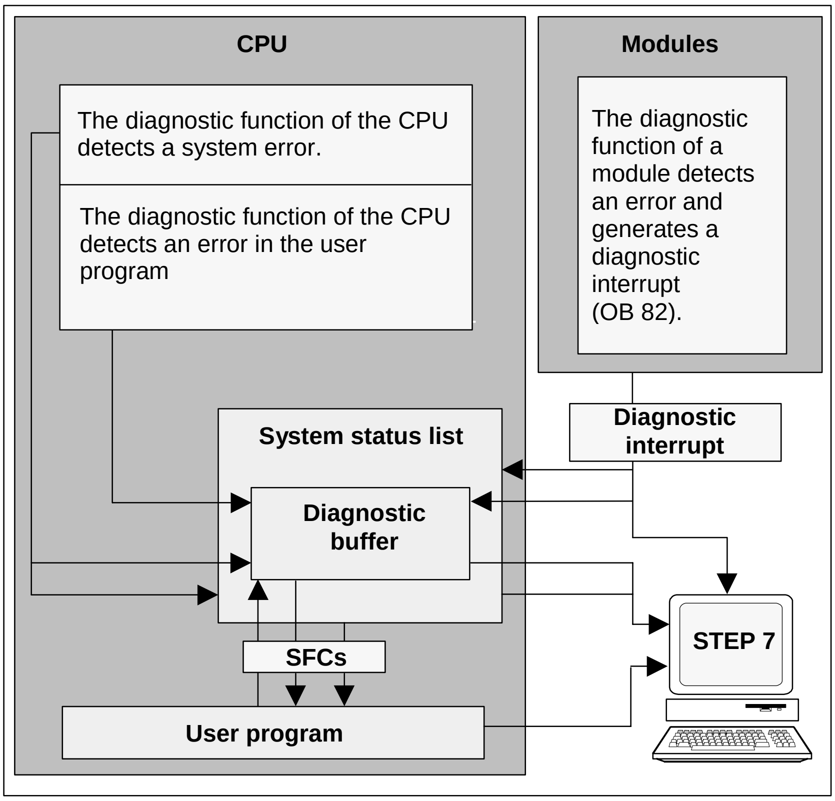
设置扫描周期

当配置硬件时，可以设置最大和最小周期。为此，在CPU/FM组态表的离线视图上双击，定义其属性。可以在“周期/时钟存储器”标签中输入适当的值。

23.8 诊断信息流

23.8.1 诊断信息流

下图给出了SIMATIC S7中诊断信息的流程。



显示诊断信息

可以在用户程序中使用SFC51 RDSYSST读取诊断条目，或者用STEP 7以通俗的语言显示诊断信息。

它们提供了以下信息：

- 出错的地点和时间
- 该条目所属的诊断事件的类型(用户自定义的诊断事件、同步/异步的错误、工作模式改变)。

生成过程控制组信息

CPU在诊断缓冲区中输入标准诊断和扩展诊断的事件。如果满足以下条件，它也会生成用于标准诊断事件的过程控制组信息：

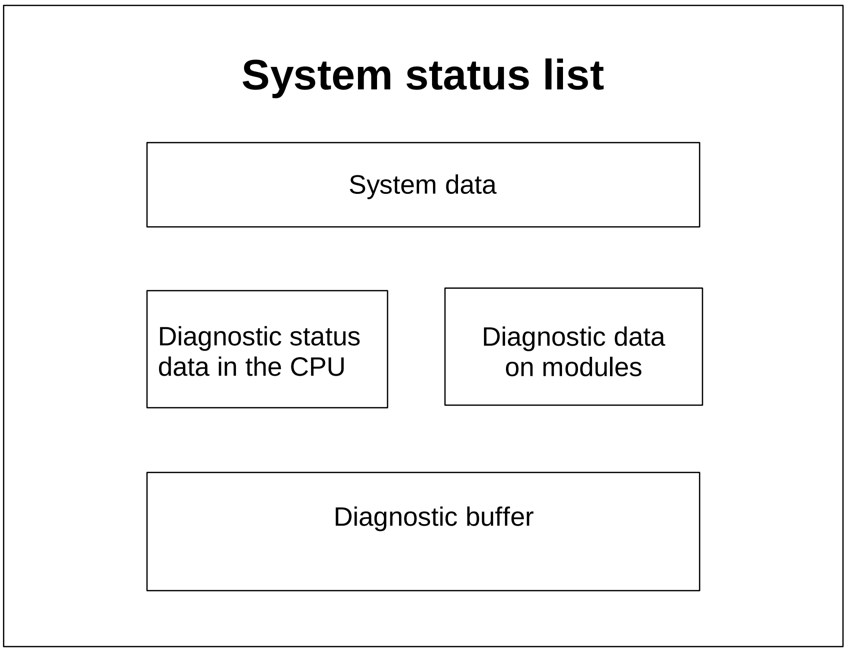
- 已经指定将要在STEP 7中生成的过程控制信息。
- 至少在CPU上为过程控制信息记录了一个显示单元。
- 在当前没有一个相应等级(总共有7个等级)的过程控制组信息时，才生成过程控制组信息。
- 每个等级可以生成一个过程控制组信息。

23.8.2 系统状态列表SSL

系统状态列表(SSL)描述可编程控制器的当前状态。它概述了组态、当前参数分配、CPU的当前状态和顺序以及属于它的模块的信息。

系统状态列表中的数据只能读取，而不能修改。它仅是一个按要求创建的虚拟列表。

可以使用系统列表显示的信息可分为四个区域。



读取系统状态列表

有两种方法可用于读取系统状态列表中的信息，如下所述：

- 隐含地，通过来自编程设备的STEP 7菜单命令(例如，存储器组态、静态CPU数据、诊断缓冲区、状态显示)。
- 明确地，通过用户程序中的系统功能SFC 51 RDSYSST，只要输入所需部分系统状态列表的编号(参见块帮助)。

系统状态列表的系统数据

系统数据是CPU固有的或者已分配特征数据。下表给出了可以显示信息的议题(部分系统状态列表)：

议题	信息
模块标识	模块的订货号、类型识别号和版本
CPU特征	CPU的时间系统、系统特性(例如， 多值计算)和语言描述
存储器区	模块的存储器组态(工作存储器的大小)。
系统区域	模块的系统存储器(例如，存储器位的数目、定时器、计数器、存储器类型)。
块类型	模块上存在哪些块(OB、DB、SDB、FC、FB)，各类块的最大数目以及块类型的最大空间
中断和错误的分配	分配给OB的中断/错误
中断状态	中断处理/生成的中断的当前状态
优先级的状态	执行哪一个OB，禁止哪一个优先级取决于参数设置
工作模式和模式转换	哪一个工作模式可行，最后的工作模式改变，当前工作模式

CPU中的诊断状态数据

诊断状态数据描述由系统诊断所监控的部件的当前状态。下表给出了可以显示信息的议题(部分系统状态列表)：

议题	信息
通讯状态数据	当前在系统中设置的所有通讯功能
诊断模块	在CPU上记录的具有诊断能力的模块
OB的启动信息列表	关于CPU的OB的启动信息
启动事件列表	启动事件和OB的优先级
模块状态信息	有关所有插入的、有故障的或生成硬件中断的已分配模块的状态信息

关于模块的诊断数据

除了CPU，还有其它具有诊断能力的模块(SM、CP、FM)，它们的数据输入到系统状态列表中。下表给出了可以显示信息的议题(部分系统状态列表)：

议题	信息
模块诊断信息	模块启动地址、内部/外部故障、通道故障、参数错误(4个字节)
模块诊断数据	特殊模块的所有诊断数据

23.8.3 发送个人诊断消息

还可以使用系统功能SFC 52 WRUSMSG来扩展SIMATIC S7的标准系统诊断：

- 在诊断缓冲区中输入个人诊断信息(例如，关于用户程序执行的信息)。
- 发送用户定义的诊断信息来登录站(监控设备，如PG、OP或TD)。

用户定义的诊断事件

诊断事件分成事件类别1至F。用户定义的诊断事件属于事件类别8至B。这些事件分成如下两组：

- 事件类别8和9包含具有固定编号和预定义文本的消息，这些消息可根据编号进行调用。
- 事件类别A和B包含可任意分配一个编号(A000 - A0FF、B000 - B0FF)以及文本的消息。

将诊断消息发送到站

除了在诊断缓冲区中生成用户定义的条目外，还可以使用SFC52 WRUSMSG发送用户自定义的诊断消息来登录显示设备。当使用SEND = 1调用SFC52时，诊断消息写入到发送缓冲区，然后自动发送到在CPU上登录的一个或多个站。

如果不能发送消息(例如，因为没有登录显示设备，或因为发送缓冲区已满)，那么依然在诊断缓冲区中输入用户定义的诊断事件。

生成带确认的消息

如果要确认一个用户定义的诊断事件，并希望记录该确认，可如下操作：

- 当事件进入事件状态时，将1写入到一个布尔型的变量中；当事件离开事件状态时，将0写入到该变量中。
- 然后，可以使用SFB33 ALARM监控该变量。

23.8.4 诊断功能

系统诊断会检测、评估以及报告可编程控制器中发生的错误。为此，具有系统诊断能力的每个CPU和每个模块(例如，FM 354)都有一个诊断缓冲区，在该缓冲区中，按照事件的发生顺序输入所有诊断事件的详细信息。

诊断事件

下列条目显示为诊断事件，例如：

- 模块上的内、外部故障
- CPU 中的系统错误
- 工作模式变化(例如，从 RUN 变为 STOP)
- 用户程序中的错误
- 插入/删除模块
- 通过系统功能 SFC52 输入的用户消息

存储器复位后，诊断缓冲区中的内容保持。通过诊断缓冲区，可以在以后分析系统中的错误，查找引起STOP的原因或追踪并归类单个诊断事件的发生。

获取诊断数据

没有必要为采集通过系统诊断取得的诊断数据而编程。这是一个自动运行的标准特性。SIMATIC S7提供各种诊断功能。一些功能集成在CPU上，另一些功能由模块提供(SM、CP和FM)。

显示故障

在模块的前面板上显示内、外部模块故障。LED显示内容及其评估在S7硬件手册中进行描述。对于S7-300，内部和外部故障作为一组错误一起显示。
CPU识别系统错误和用户程序中的错误，并在系统状态列表和诊断缓冲区中输入诊断消息。这些诊断消息可在编程设备上读取。
具有诊断能力的信号和功能模块会检测内部和外部模块错误，并生成一个可通过中断OB进行响应的诊断中断。

23.9 处理错误的程序措施

当CPU检测到程序处理中的错误(同步错误)以及可编程控制器中的错误(异步错误)时，就调用处理该错误的相应组织块(OB)：

错误	错误OB
I/O冗余错误	OB70
CPU冗余错误	OB72
时间错误	OB80
电源错误	OB81
诊断中断	OB82
插入/删除模块中断	OB83
CPU硬件故障	OB84
优先级错误	OB85
机架故障或分布式I/O中的站故障	OB86
通讯错误	OB87
编程错误	OB121
I/O访问错误	OB122

如果没有合适的OB可供使用，那么CPU进入停止模式。否则，可以在OB中存储如何响应该错误情况的指令。这表示可以减少或消除错误影响。

基本步骤

创建并打开OB

1. 显示CPU的模块信息。
2. 选择“性能数据”标签。
3. 根据所显示的列表，确定是否允许该CPU使用要编程的OB。
4. 在程序的“块”文件夹中插入OB，然后打开该OB。
5. 进入进行错误处理的程序。
6. 将OB下载到可编程控制器。

处理错误的程序措施

1. 评估OB的局部数据，确定引起错误的确切原因。
   局部数据中的变量OB8xFLTID和OB12xSWFLT包含错误代码。在“系统和标准功能参考手册”中描述了它们的含义。
2. 跳转到响应该错误的程序段。

在系统和标准功能的在线参考帮助中的标题为“SFC51 (RDSYSST)模块诊断实例”的文本中，可以获取处理诊断中断的实例。

在相应的关于块的帮助中，可以获取关于OB、SFB和SFC的详细资料。

23.9.1 评估输出参数RET_VAL

利用RET_VAL输出参数(返回值)，系统功能指示CPU是否可以正确执行SFC功能。

返回值中的错误信息

返回值为整型数据类型(INT)。整数符号指示它是正整数还是负整数。返回值与“0”值之间的关系指示在执行该功能期间是否发生错误(参见表)：

- 如果在执行该功能期间，发生错误，那么返回值小于“0”。整数的符号位为“1”。
- 如果执行该功能期间，无错误，那么返回值大于或等于“0”。整数的符号位为“0”。

由CPU处理SFC	返回值	整数符号
发生错误	小于“0”	负(符号位为“1”)
无错误	大于或等于“0”	正(符号位为“0”)

响应错误信息

如果在执行SFC期间，发生错误，那么SFC在返回值(RET_VAL)中提供一个错误代码。

区别下列各种情况：

- 所有SFC都可输出的常规错误代码，以及
- SFC根据其特殊功能可输出的特殊错误代码。

传送功能值

一些SFC还使用输出参数RET_VAL来传送功能值，例如，SFC64 TIMETCK使用RET_VAL传送所读取的系统时间。

在SFB/SFC帮助中，可以获取关于输出参数RET_VAL的更多详细资料。

23.9.2 对检测到错误响应的错误OB

可检测错误

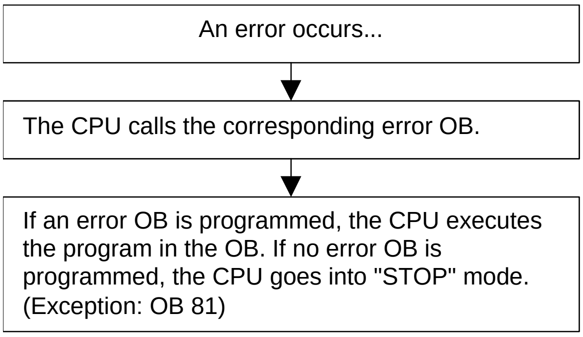
系统程序可检测下列错误：

- CPU功能异常
- 系统程序执行错误
- 用户程序错误
- I/O错误

根据错误类型，CPU进入停止模式或调用错误OB。

编程响应

可以设计程序来响应不同类型的错误，以及确定CPU的响应方式。处理特殊错误的程序可以保存在错误OB中。.如果调用了错误OB，那么执行该程序。



错误OB

区别同步错误和异步错误的方法如下：

- 同步错误可分配给MC7指令(例如，给已经删除的信号模块的加载指令)。
- 异步错误可分配给优先级或整个可编程逻辑控制器(例如，超出周期)。

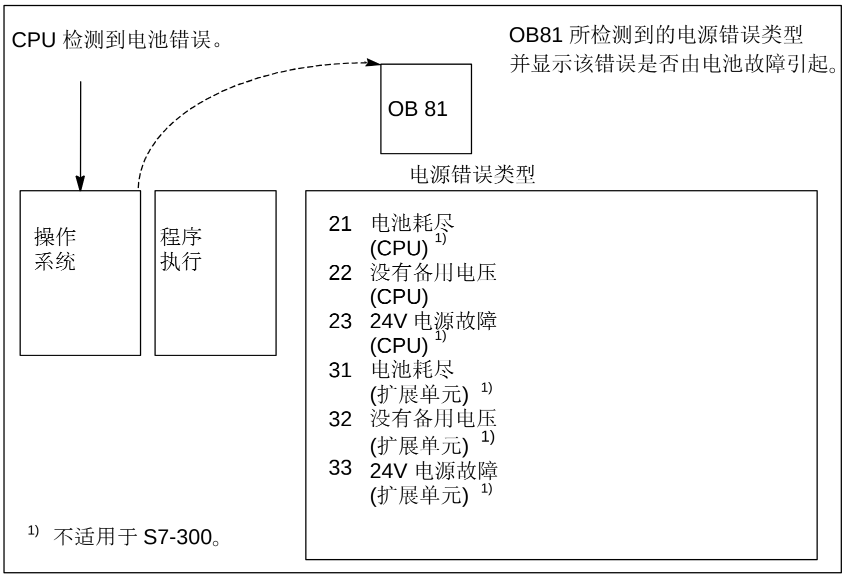
下表显示了可能发生的错误类型。请参见“S7-300可编程控制器、硬件和安装手册”或“S7-400、M7-400可编程控制器、硬件和安装手册”，获取CPU是否提供所指定的OB的信息。

错误类别	错误类型	OB	优先级
冗余	I/O冗余错误(仅在H CPU中)	OB 70	25
	CPU冗余错误(仅在H CPU中)	OB 72	28
异步	时间错误	OB 80	26
	电源错误	OB 81	(如果在启动程序中调用错误
	诊断中断	OB 82	OB，则为28)
	插入/删除模块中断	OB 83	
	CPU硬件故障	OB 84	
	程序顺序错误	OB 85	
	机架故障	OB 86	
	通讯错误	OB 87	
同步	编程错误	OB 121	引起错误的OB的优先级
	I/O访问错误	OB 122	

使用错误OB81的实例

利用错误OB的局部数据(启动信息)，可以评估已经发生的错误类型。

例如，如果CPU检测到电池错误，那么操作系统调用OB81 (参见图)。



可以编写一个程序，评估由OB81调用触发的事件代码。还可以编写一个产生响应的程序，如激活一个与操作员站上的灯相连的输出。

错误OB81的局部数据

下表显示了在本例中，OB81的变量声明表中必须声明的临时变量。

电池错误(BOOL)符号必须看作一个输出(例如，Q 4.0)，以便程序的其它部分可以访问这些数据。

声明	名称	类型	描述
TEMP	OB81EVCLASS	BYTE	错误类别/错误标识符39xx
TEMP	OB81FLTID	BYTE	错误代码： b#16#21 = 至少有一个CPU的备用电池耗尽[1)] b#16#22 = CPU中没有备用电压 b#16#23 = CPU中24V电源故障 [1] b#16#31 = 至少有一个扩展基架的备用电池耗尽[1)] b#16#32 = 扩展机架中没有备用电压 [1] b#16#33 = 扩展机架中24V电源故障[1)]
TEMP	OB81PRIORITY	BYTE	优先级= 26/28
TEMP	OB81OBNUMBR	BYTE	81 = OB81
TEMP	OB81RESERVED1	BYTE	保留
TEMP	OB81RESERVED2	BYTE	保留
TEMP	OB81MDLADDR	INT	保留
TEMP	OB81RESERVED3	BYTE	只与错误代码B#16#31、B#16#32、B#16#33有关
TEMP	OB81RESERVED4	BYTE	
TEMP	OB81RESERVED5	BYTE	
TEMP	OB81RESERVED6	BYTE	
TEMP	OB81DATETIME	DATEAND TIME	启动OB时的日期和时间
[1)] =不适用于S7-300。			

错误OB81的实例程序

STL实例程序显示了如何在OB81中读取错误代码。

程序结构如下：

- 读取OB81中的错误代码(OB81FLTID)，然后与事件“电池耗尽”(B#16#3921)的值进行比较。
- 如果错误代码与“电池耗尽”的代码一致，那么程序跳转到Berr标签，然后激活输出*batteryerror*。
- 如果错误代码与“电池耗尽”的代码不一致，那么程序将该代码与“电池故障”的代码进行比较。
- 如果错误代码与“电池故障”的代码一致，那么程序跳转到Berr标签，然后激活输出*batteryerror*。否则，终止该块。

AWL		描述
L	B#16#21	// 比较事件代码“电池耗尽”
		// (B#16#21)和
L	#OB81_FLT_ID	// OB81的错误代码。
==I		// 如果相同(电池已耗尽)，
		// 跳转到Berr。
JC Berr		
L	B#16#22	// 比较事件代码“电池故障”
		// (b#16#22)和
==I		// 和OB81的错误代码。
JC BF		// 如果相同，跳转到Berr。
BEU		// 没有关于电池故障的消息
Berr:L	B#16#39	// 比较下一个事件的ID和
L	#OB81_EV_CLASS	// OB81的错误代码。
==I		// 如果发现发生电池故障或电池耗尽，
//		
S	batteryerror	// 那么设置输出“电池错误”。
//(符号表中的变量)		
L	B#16#38	// 比较结束事件的ID
==I		// 和OB81的错误代码。
R	batteryerror	// 当修复该错误时，
		// 复位输出“电池错误”。

在相应的关于块的帮助中，可以获取OB、SFB和SFC的详细资料，以及事件标识符的解释。

23.9.3 插入用于错误检测的替换值

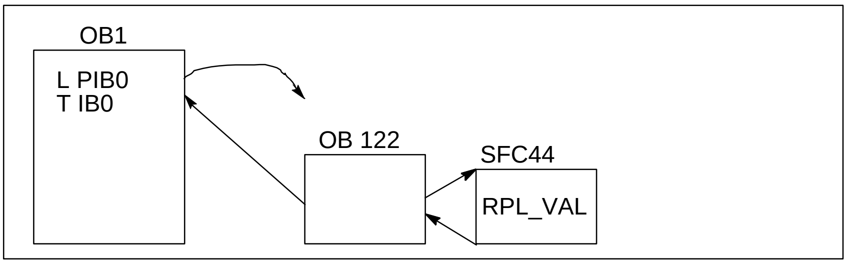
对于特定类型的错误(例如，影响输入信号的断线)，可以给由于错误而导致不可用的值提供替换值。有两种提供替换值的方法：

- 可以使用STEP 7，将替换值分配给可组态输出模块。不能分配参数的输出模块具有缺省的替换值0。
- 通过SFC44 RPLVAL，可以在错误OB中编程替换值(仅适用于输入模块)。

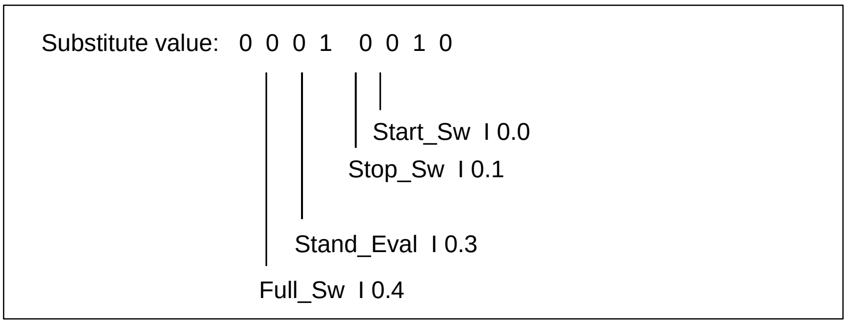
对于导致同步错误的所有加载指令，可以在错误OB中为累加器内容指定替换值。

用于替换值的实例程序

在下列实例程序中，在SFC44 RPLVAL中可以使用替换值。下图显示了当CPU识别出输入模块没有响应时，如何调用OB122。



在该实例中，在程序中输入下图中的替换值，从而程序可采用可行值继续运行。



如果输入模块发生故障，那么语句L PIB0的处理产生一个同步错误，并启动OB122。标准情况下，加载指令读取值0。然而，对于SFC44，可以定义任何适合该过程的替换值。SFC用指定的替换值替换累加器内容。

可以在OB122中编写下列实例程序。下表显示了在本例中，OB122的变量声明表中必须声明的临时变量。

声明	名称	类型	描述
TEMP	OB122EVCLASS	BYTE	错误类别/错误标识符29xx
TEMP	OB122SWFLT	BYTE	错误代码： 16#42、16#43、16#44 [1]、16#45 [1]
TEMP	OB122PRIORITY	BYTE	优先级=发生错误处的OB的优先级
TEMP	OB122OBNUMBR	BYTE	122 = OB122
TEMP	OB122BLKTYPE	BYTE	发生错误处的块类型
TEMP	OB122MEMAREA	BYTE	存储区和访问类型
TEMP	OB122MEMADDR	WORD	发生错误处的存储器地址
TEMP	OB122BLKNUM	WORD	发生错误处的块编号
TEMP	OB122PRGADDR	WORD	产生错误的指令的相对地址
TEMP	OB122DATETIME	DATEANDTIME	启动OB时的日期和时间
TEMP	Error	INT	保存SFC44的错误代码
[1]不适用于S7-300。			

STL	描述
L b#16#2942 L #OB122SWFLT ==I JC Aerr	比较OB122的事件代码和读取I/O时出现的时间错误确认的事件代码(B#16#2942)。如果相同，那么跳转到“Aerr”。
L b#16#2943 <> I JC Stop	比较OB122的事件代码和寻址错误(写入不存在的模块)的事件代码(B#16#2943)。如果不相同，那么跳转到“Stop”。
Aerr: CALL ”REPL_VAL" VAL := DW#16#2912 RETVAL := #Error	“Aerr”标签：将DW#16#2912 (二进制10010)传送到SFC44 (REPL_VAL)。SFC44在累加器1中加载该值(并替换触发OB122调用的值)。在#Error中保存SFC错误代码。
L #Error L 0 ==I BEC	比较#Error和0，(如果相同，那么执行OB122时没有发生错误。如果没有错误，那么结束块。
Stop: CALL ”STP"	“Stop”标签：调用SFC46 “STP”，并将CPU变成停止模式。

23.9.4 I/O冗余错误(OB70)

描述

如果在PROFIBUS DP上发生丢失冗余(例如，在激活的DP主站上发生总线故障或在DP从站接口模块上发生错误)，或通过切换I/O从DP从站切换到激活的DP主站，那么H CPU的操作系统调用OB70。

编程OB70

必须使用STEP 7在S7程序中将OB70创建为对象。在所生成的块中编写将要在OB70中执行的程序，然后将其作为用户程序的一部分下载到CPU中。

例如，可以将OB70用于下列目的：

- 评估OB70的启动信息，并确定哪个事件会触发I/O丢失冗余。
- 用SFC51 RDSYSST (SZLID=B#16#71)来确定系统状态。

如果发生I/O冗余错误，且没有编程OB70，那么CPU不会变成停止模式。

如果下载了OB70，且H系统未处于冗余模式，则两个CPU中都会处理OB70。H系统仍然处于冗余模式。

在相应的关于块的帮助中，可以获取关于OB、SFB和SFC的详细资料。

23.9.5 CPU冗余错误(OB72)

描述

如果发生下列其中一件事情，那么H CPU的操作系统调用OB72：

- CPU丢失冗余
- 比较错误(例如，RAM、PIQ)
- 备用主站切换
- 同步错误
- SYNC子模块错误
- 更新过程失败
- 伴随启动事件之后，由位于运行模式或启动模式的所有CPU执行OB72。

编程OB72

必须使用STEP 7在S7程序中将OB72创建为对象。在所生成的块中编写将要在OB72中执行的程序，然后将其作为用户程序的一部分下载到CPU中。

例如，可以将OB72用于下列目的：

- 评估OB72的启动信息，并确定哪个事件触发CPU丢失冗余。
- 用SFC51 RDSYSST (SZLID=B#16#71)来确定系统状态。
- 响应CPU丢失冗余，尤其是设备引起的。

如果发生CPU冗余错误，并且没有编程OB72，那么CPU不会变成停止模式。
在相应的关于块的帮助中，可以获取关于OB、SFB和SFC的详细资料。

23.9.6 时间错误(OB80)

描述

发生时间错误时，CPU操作系统调用OB80。时间错误包括下列各项，例如：

- 超出最大周期
- 通过向前调整时间，跳过时间中断
- 处理优先级时，时延太大

编程OB80

必须使用STEP 7在S7程序中将OB80创建为对象。在所生成的块中编写将要在OB80中执行的程序，然后将其作为用户程序的一部分下载到CPU中。

例如，可以将OB80用于下列目的：

- 评估OB80的启动信息，并确定跳过哪个时间中断。
- 通过SFC29 CANTINT，可以取消激活跳过的时间中断，从而不执行该中断，只执行与新时间有关的时间中断。

如果在OB80中没有取消激活跳过的时间中断，那么执行第一个跳过的时间中断，并忽略所有其它中断。

如果没有编程OB80，那么当检测到时间错误时，CPU变成停止模式。
在相应的关于块的帮助中，可以获取关于OB、SFB和SFC的详细资料。

23.9.7 电源错误(OB81)

描述

如果在CPU或扩展单元中下列其中一个发生故障，那么CPU操作系统调用OB81。

- 24V电源
- 电池
- 备用系统

消除故障后也调用该OB (事件出现和消失时都调用该OB)，。

编程OB81

必须使用STEP 7在S7程序中将OB81创建为对象。在所生成的块中编写将要在OB81中执行的程序，然后将其作为用户程序的一部分下载到CPU中。

例如，可以将OB81用于下列目的：

- 评估OB81的启动信息，并确定已经发生哪些电源错误。
- 查明具有故障电源的机架数目。
- 激活操作员站上的灯，以指示维护人员应该更换电池。

如果不编程OB81，则当检测到电源错误时，与所有其它异步错误OB相反，CPU不进入STOP模式。不过，该错误会输入诊断缓冲区，并且前面板上的相应LED会指示该错误。

在相应的关于块的帮助中，可以获取关于OB、SFB和SFC的详细资料。

23.9.8 诊断中断(OB82)

描述

当具有诊断能力并启用诊断中断的模块检测到错误，以及消除错误时，CPU操作系统调用OB82 (事件出现或消失时，调用该OB)。

编程OB82

必须使用STEP 7在S7程序中将OB82创建为对象。在所生成的块中编写将要在OB82中执行的程序，然后将其作为用户程序的一部分下载到CPU中。

例如，可以将OB82用于下列目的：

- 评估OB82的启动信息。
- 获取已发生错误的精确诊断信息。

触发诊断中断时，发生故障的模块自动在诊断中断OB的启动信息以及诊断缓冲区中输入4字节的诊断数据以及它们的启动地址。这可为您提供错误发生时间和错误所在模块的信息。

通过使用OB82中的合适程序，可以进一步评估模块的诊断数据(在哪个通道上发生错误，发生何种错误)。通过SFC51 RDSYSST，可以读取模块诊断数据，并使用SFC52 WRUSRMSG在诊断缓冲区中输入该信息。还可以将用户定义的诊断信息发送到监控设备。

如果没有编程OB82，那么触发诊断中断时，CPU会变成停止模式。
在相应的关于块的帮助中，可以获取关于OB、SFB和SFC的详细资料。

23.9.9 插入/删除模块中断(OB83)

描述

S7-400 CPU以1秒左右的间隔时间监控中央机架和扩展机架中模块的出现。

接通电源后，CPU检查通过STEP 7创建的组态表中所列的所有模块是否都确实已经插入。如果出现所有模块，那么保存实际组态，并且该组态用作循环监控模块的参考值。在每个扫描周期中，最新检测到的实际组态与以前的实际组态进行比较。如果组态之间存在差异，那么发出插入/删除模块中断信号，并在诊断缓冲区和系统状态列表中生成一个条目。在运行模式下，插入/删除模块中断OB启动。

注意

禁止在运行模式下删除电源模块、CPU和IM。

删除和插入模块之间，必须间隔至少2秒的时间，使CPU可以检测到已经删除或插入一个模块。

将参数分配给新插入的模块

如果在运行模式下插入一个模块，那么CPU检查新模块的模块类型是否与原模块类型相匹配。如果类型匹配，那么给该模块分配参数。缺省参数或通过STEP 7分配的参数会传送至该模块。

编程OB83

必须使用STEP 7在S7程序中将OB83创建为对象。在所生成的块中编写将要在OB83中执行的程序，然后将其作为用户程序的一部分下载到CPU中。

例如，可以将OB83用于下列目的：

- 评估OB83的启动信息。
- 通过系统功能SFC55 - 59，将参数分配给新插入的模块。

如果没有编程OB83，那么发生插入/删除模块中断时，CPU从运行模式变成停止模式。

在相应的关于块的帮助中，可以获取关于OB、SFB和SFC的详细资料。

23.9.10 CPU硬件故障(OB84)

描述

当在MPI网络接口、通讯总线接口或分布式I/O的网卡接口上检测到错误时，CPU操作系统调用OB84；例如，在线路上检测到错误信号电平时。消除故障时也调用该OB (事件出现和消失时都调用该OB)。

编程OB84

必须使用STEP 7在S7程序中将OB84创建为对象。在所生成的块中编写将要在OB84中执行的程序，然后将其作为用户程序的一部分下载到CPU中。

例如，可以将OB84用于下列目的：

- 评估 OB84 的启动信息。
- 通过系统功能 SFC52 WRUSMSG，将消息发送到诊断缓冲区。

如果没有编程OB84，那么当检测到CPU硬件故障时，CPU变成停止模式。
在相应的关于块的帮助中，可以获取关于OB、SFB和SFC的详细资料。

23.9.11 程序顺序错误(OB85)

描述

在下列情况下，CPU操作系统调用OB85：

- 存在中断 OB 的启动事件，但由于还没有将该 OB 下载到 CPU 而不能执行该 OB。
- 访问系统功能块的背景数据块时发生错误。
- 更新过程映像表时发生错误(模块不存在或处于故障状态)。

编程OB85

必须使用STEP 7在S7程序中将OB85创建为对象。在所生成的块中编写将要在OB85中执行的程序，然后将其作为用户程序的一部分下载到CPU中。

例如，可以将OB85用于下列目的：

- 评估 OB85 的启动信息，并确定哪个模块处于故障状态或没有插入(指定模块启动地址)。
- 通过 SFC49 LGCGADR，查找所涉及的模块的插槽。

如果没有编程OB85，那么当检测到优先级错误时，CPU变成停止模式。
在相应的关于块的帮助中，可以获取关于OB、SFB和SFC的详细资料。

23.9.12 机架故障(OB86)

描述

如果没有编程OB86，那么当检测到机架故障时，CPU进入STOP模式；例如：

- 机架故障(IM丢失或出故障或连接电缆断裂)
- 机架上发生分布式电源故障
- 在SINEC L2DP总线系统的主站系统中，DP从站发生故障

消除故障时也调用该OB (事件出现和消失时都调用该OB)。

编程OB86

必须使用STEP 7在S7程序中将OB86创建为对象。在所生成的块中编写将要在OB86中执行的程序，然后将其作为用户程序的一部分下载到CPU中。

例如，可以将OB86用于下列目的：

- 评估OB86的启动信息，并确定哪个机架处于故障状态或丢失。
- 通过系统功能SFC 52 WRUSMSG在诊断缓冲区中输入消息，并将该消息发送到监控设备。

如果没有编程OB86，那么当检测到机架故障时，CPU变成停止模式。
在相应的关于块的帮助中，可以获取关于OB、SFB和SFC的详细资料。

23.9.13 通讯错误(OB87)

描述

当使用通讯功能块进行数据交换或在全局数据通讯期间发生通讯错误时，CPU操作系统调用OB87，例如：

- 接收到全局数据时，检测到错误帧标识符。
- 全局数据的状态信息的数据块不存在或太短。

编程OB87

必须使用STEP 7在S7程序中将OB87创建为对象。在所生成的块中编写将要在OB87中执行的程序，然后将其作为用户程序的一部分下载到CPU中。

例如，可以将OB87用于下列目的：

- 评估OB87的启动信息。
- 在丢失全局数据通讯状态信息的数据块时，用于创建该数据块。

如果没有编程OB87，则当检测到通讯错误时，CPU进入STOP模式。

在相应的关于块的帮助中，可以获取关于OB、SFB和SFC的详细资料。

23.9.14 编程错误(OB121)

描述

发生编程错误时，CPU操作系统调用OB121，例如：

- 已寻址的定时器不存在。
- 没有加载所调用的块。

编程OB121

必须使用STEP 7在S7程序中将OB121创建为对象。在所生成的块中编写将要在OB121中执行的程序，然后将其作为用户程序的一部分下载到CPU中。

例如，可以将OB121用于下列目的：

- 评估OB121的启动信息。
- 在消息数据块中输入错误原因。

如果没有编程OB121，那么当检测到编程错误时，CPU变成停止模式。

在相应的关于块的帮助中，可以获取关于OB、SFB和SFC的详细资料。

23.9.15 I/O访问错误(OB122)

描述

当STEP 7指令访问在最后一次热重启动时没有分配模块的信号模块的输入或输出时，CPU操作系统调用OB122，例如：

- 直接I/O访问错误(模块故障或丢失)
- 访问CPU未知的I/O地址。

编程OB122

必须使用STEP 7在S7程序中将OB122创建为对象。在所生成的块中编写将要在OB122中执行的程序，然后将其作为用户程序的一部分下载到CPU中。

例如，可以将OB122用于下列目的：

- 评估OB122的启动信息。
- 调用系统功能SFC 44，给输入模块提供一个替换值，从而程序可以采用有意义的、与过程有关的值继续执行。

如果没有编程OB122，那么当检测到I/O访问错误时，CPU变成停止模式。

在相应的关于块的帮助中，可以获取关于OB、SFB和SFC的详细资料。

24 打印和归档

24.1 打印项目文档

一旦为自动化任务创建了程序，就可以使用集成在STEP 7中的打印功能来打印输出项目文档的所有重要数据。

可以打印的项目部分

可以从SIMATIC管理器直接打印对象内容，也可以打开相关对象，然后启动打印过程进行打印。

通过SIMATIC管理器可以直接打印项目的下列部分：

- 对象树(项目/库的结构)
- 对象列表(对象文件夹的内容)
- 对象内容
- 消息

通过打开相关项目，可以打印项目的下列部分：

- 以梯形图、语句表或功能块图或以其它语言(可选软件)表示的块
- 带有绝对地址符号名的符号表
- 可编程控制器中的模块排列的组态表和模块参数
- 诊断缓冲区内容
- 带有监视格式、监视和修改值的变量表
- 引用数据；例如交叉索引表、赋值表、程序结构、未使用地址表、无符号地址表
- 全局数据表
- 带有模块状态的模块信息
- 操作员相关文本(用户文本和文本库)
- 可选包的文档，如其它编程语言的文档

DOCPRO可选包

要创建、编辑和打印标准化的接线手册，可使用可选软件包DOCPRO。这就创建了满足DIN和ANSI标准的设备文档。

24.1.1 打印的基本步骤

要进行打印，可如下操作：

1. 打开相应的对象，在屏幕上显示要打印的信息。
2. 在应用程序窗口中，使用菜单命令**文件 > 打印**打开“打印”对话框。根据当前所在的应用程序，菜单栏中出现的第一个条目可能不是“文件”，而是该应用程序处理的对象，例如“符号表”。
3. 如有必要，可以在对话框中更改打印选项(打印机、打印范围、打印份数等)，然后关闭该对话框。

某些对话框具有“打印”按钮，例如，“模块信息”对话框。点击该按钮可以打印对话框的内容。

无需打开块，就可以在SIMATIC管理器中，使用菜单命令**文件 > 打印**直接打印。

24.1.2 打印功能

在对打印对象进行打印时，下列附加功能均可使用：

打印对象	菜单命令	功能	功能	功能
		打印预览	页面设置、“页面格式”标签	页面设置、“页眉和页脚”标签
块、STL源文件	文件 > *	●	●	●
模块信息		–	●	●
全局数据表	GD表>*	●	●	●
组态表	站>*	●	●	●
对象、对象文件夹	文件 > *	–	●	●
引用数据	引用数据 > *	●	●	●
符号表	表格 > *	●	●	●
变量表	表格 > *	–	●	●
连接表	网络 > *	●	●	●
与操作员相关的文本(用户文本、文本库)	文本 > *	●	●	●
* : * 符号用作各自功能菜单命令的通配符(例如打印预览或页面设置)				

对单个打印作业进行打印的逐步说明可在下面找到：
如何打印。

打印预览

您可使用“打印预览”功能来显示要打印的文档的页面布局。

如果该文档包含多个页面，则在页面右下角的页码后出现两个句点。最后一页没有这些点号，说明后面不再有页面。

注意

已完成的文档的打印格式将不会显示在打印预览中。

设置页面格式和页眉与页脚

您可使用**文件 > 页面设置**菜单命令设置您想要打印的所有文档的纸张大小(例如A4、A5、Letter)和页面格式以及方向(纵向或横向)。.此外，您可选择是将设置值应用于整个项目还是仅应用于当前章节。

调整文档布局以便其与要求的纸张格式相匹配。如果文档太宽，则右边的页边将打印在相邻的页面上。

如果您选择了具有页边距的页面格式(例如，A4的页边距)，则所打印的文档在页面的左边将留有页边距，您可将其穿孔，以进行装订。

为了在希望打印的文档的整个项目中或只是当前章节中设置页眉和页脚，可转到“标签区域”标签。如果该文档包含多个页面，则在页面右下角的页码后出现两个句点。最后一页没有这些点号，说明后面不再有页面。这是检查打印输出是否完整的一种快速方法。在打印预览中也可以看见这两个句点。

24.1.3 打印对象树时的特殊注意事项

在“打印对象列表”对话框中，除了对象列表外，您还可以通过选择“树窗口”选项来打印对象树。

如果在“打印范围”中选择了选项“全部”，那么将打印整个树结构。如果选择了选项按钮“选择”，那么将打印所选对象往下的树结构。

注意

在该对话框中所作的设置仅适用于打印列表或树，并不适用于打印对象的内容；相关应用程序中的设置用于打印对象的内容。

24.2 对项目和库进行归档

24.2.1 对项目和库进行归档

您可将单个项目或库以压缩的形式存储在一个归档文件中。该压缩存储过程可在硬盘或便携式数据介质(例如软盘)上进行。

归档程序

在STEP 7中，您可选择适合的归档程序。归档程序ARJ和PKZIP 4.0均是STEP 7软件包的一部分。这些程序及其描述均位于其安装路径文件夹...\Step7\S7bin\中。

如果使用下列的归档程序，那么，您将需要所述的版本(或更新的版本)：

- PKZip Commandline V4.0 (随STEP 7提供)
- 版本6.0以上的WinZip
- 版本1.02以上的JAR
- ARJ V2.4.1a (仅适用于检索归档，已随STEP 7提供)
- ARJ32 V3.x (仅适用于检索归档)
- 版本2.13以上的LHArc (仅适用于检索归档)

特殊事项

至于STEP 7 V5.2，仅支持归档程序PKZip 4.0、JAR、WinZip。然而，上面所列出的其它程序均支持检索。

如果，在早期版本的STEP 7中，使用了程序ARJ32 V3.x创建文档，那么，这些文档将只能使用同一个程序进行检索。

与本地驱动器相比，在网络驱动器上使用PKZIP V4.0创建文档需要的时间要多得多。

24.2.2 用于保存/归档

另存为

使用该功能，可以创建项目**副本**并以其它名称保存。
使用该功能可以：

- 创建备份副本
- 复制已存在的项目以用于其它目的。

要使用创建副本的最快方法，可在对话框中选择不用重新排列的“另存为”选项。将不作检查就复制从项目目录往下的整个文件结构，然后以其它名称保存。
数据介质必须有足够的空间来存储备份副本。不要尝试将项目保存到磁盘，因为磁盘通常没有足够的可用空间。要往磁盘上传输项目数据，使用“归档”功能。

带有重新排列任务的保存将花费较长的时间，但是它会在不能复制和保存对象时，显示一条消息。出现该情况的原因可能是缺少可选包或对象数据损坏。

归档

可以将单个项目或库以压缩形式存储在一个归档文件中。该压缩存储过程可在硬盘或便携式数据介质(例如软盘)上进行。
只能以归档文件的形式往磁盘上传输项目。如果项目太大，那么选择一个可以创建跨磁盘归档的归档程序。
不能对压缩到归档文件中的项目或库进行编辑。如果希望再次编辑它们，必须提取数据包，即恢复项目或库。

24.2.3 归档要求

要对项目或库进行归档，必须满足下列要求：

- 必须在系统中安装归档程序。在“归档/恢复过程”的在线帮助主题中解释了与 STEP 7 的链接。
- 项目的所有数据必须一律位于项目目录或项目的子目录中。在 C 开发环境中工作时，可以在其它位置存储数据。那么，归档文件中不包含这些数据。
- 至于 STEP 7 V5.2，仅支持归档程序 PKZip 4.0、JAR、WinZip。但是，在恢复时，还支持 ARJ 和 LHArc 程序。

24.2.4 归档/恢复过程

使用菜单命令**文件 > 归档**或**文件 > 恢复**来归档/恢复项目或库。

注意

不能对压缩到归档文件中的项目或库进行编辑。如果希望再次编辑它们，必须提取数据，即恢复项目或库。

恢复时，自动在项目/库列表中包含所恢复的项目或库。

设置目标目录

要设置目标目录，在SIMATIC管理器中使用菜单命令**选项 > 自定义**来打开“自定义”对话框。

在该对话框的“归档”标签中，可以打开或关闭选项“恢复时检查目标路径”。

如果该选项无效，那么在同一个对话框的“通用”标签中为“项目存储位置”和“库存储位置”而设置的路径将用来作为恢复的目标路径。

将归档文件复制到磁盘

可以对项目/库进行归档，然后将归档文件复制到磁盘。也可以在“归档”对话框中选择软盘驱动器作为目标目录。

25 使用M7可编程控制系统

25.1 M7系统的步骤

M7-300/M7-400自动化计算机的标准PC结构构成了SIMATIC自动化平台的自由可编程扩展。您可以使用高级语言，例如C语言或使用图形化语言CFC (持续功能表)来为SIMATIC M7用户程序编程。

要创建程序，除了STEP 7以外，还需要M7-300/400的系统软件M7-SYS RT和M7程序的开发环境(ProC/C++或CFC)。

基本步骤

当您使用SIMATIC M7创建自动化解决方案时，您将面对一系列的基本任务。下表给出了大多数项目都需要执行的任务，并将其分配给一个基本过程。该表也可以作为本手册或其它手册相关章节的参考。

步骤	描述
设计自动化解决方案	M7专用； 参考： M7-SYS RT编程手册
启动STEP 7	对于S7
创建项目结构 安装站 组态硬件	对于S7
组态通讯连接	对于S7
定义符号表	对于S7
创建C或CFC用户程序	M7专用； 参考：ProC/C++
组态操作系统 在M7-300/M7-400上安装操作系统 下载硬件配置和用户程序到M7	M7专用； 参考： M7-SYS RT用户手册
测试和调试用户程序	ProC/C++
监视操作和M7诊断	对于S7，但没有自定义的诊断
打印和归档	对于S7

在M7中有何不同？

对于M7-300/M7-400，下列功能在STEP 7中不支持：

- 多计算- 同步操作多个CPU
- 强制变量
- 全局数据通讯
- 自定义诊断

管理M7可编程控制系统

通过M7可编程控制系统上的下列任务，STEP 7为您提供了特殊的支持：

- 在M7-300/M7-400上安装操作系统
- 通过编辑系统文件组态操作系统
- 下载用户程序到M7-300/M7-400
- 更新固化程序

要访问M7可编程控制系统的管理，选择M7程序文件夹后，在包含M7 CPU或FM的站的项目中，打开关联菜单，从中选择菜单命令：

PLC > 管理M7系统

您可以从在线帮助和M7-SYS RT用户手册中获取详细的指导。

25.2 M7编程的可选软件

M7可选软件

STEP 7提供了完成下列工作所需的基本功能：

- 创建并管理项目
- 硬件配置及硬件参数设定
- 网络和网络连接组态
- 管理符号数据

无论您是否使用SIMATIC S7或SIMATIC M7可编程控制器，都提供这些功能。

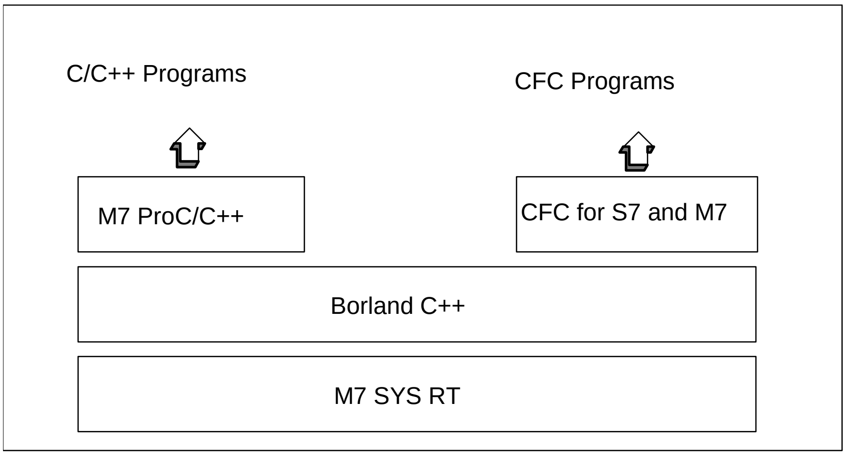
为了创建M7应用程序，除STEP 7外，您还需要M7可选软件。

软件	内容
M7-SYS RT	• M7 RMOS32操作系统 • M7API系统库 • 支持MPI
用于S7和M7的CFC	用于CFC (连续功能图)程序的编程软件
M7-ProC/C++	• STEP 7中的Borland开发环境链接 • 符号导入编辑器和发生器 • Organon xdb386高级语言调试工具
Borland C++	Borland C/C++开发环境

结合M7可选软件，STEP 7可支持下列附加任务：

- 通过多点通讯接口(MPI)，将数据下载到M7可编程控制系统
- 查询M7可编程控制系统的信息
- 完成M7可编程控制系统的特殊设置，以及复位M7

下图所示为用于M7编程的M7可选软件的相互关系。



结论

要创建的对象	需要的**M7**软件选项
C/C++程序	• M7-SYS RT • M7-ProC/C++ • Borland C++
CFC程序	• M7-SYS RT • 用于S7和M7的CFC • Borland C++

软件和软件所支持的任务

创建M7应用程序所需要的特殊工具，一部分集成在STEP 7中，一部分集成在M7可选软件中。

下表说明各种软件包所支持的任务：

软件	所提供的支持
STEP 7	• 安装M7操作系统 • 管理M7可编程控制系统 • 下载、启动及删除M7程序 • 显示状态和诊断数据 • 复位CPU
M7-SYS RT	M7操作系统和M7系统应用软件有助于实现下列任务： • 控制程序处理 • 管理存储器和资源 • 访问计算机硬件和SIMATIC硬件 • 处理中断 • 诊断 • 状态监视 • 通讯
M7-ProC/C++	• 集成代码创建(将Borland开发环境集成到STEP 7中) • 将项目的符号链接到源代码中 • 通过集成的调试功能
Borland C++	• 创建C和C++程序
用于S7和M7的CFC	• 创建、测试和调试CFC程序 • 启动和运行CFC程序

25.3 M7-300/M7-400操作系统

对于用C和C++高级语言创建的应用程序来说，操作系统提供的这些应用软件是至关重要的。操作系统为应用程序完成下列任务：

- 访问硬件
- 管理资源
- 系统集成
- 与系统中的其它组件通讯

为了执行自动化任务，SIMATIC M7自动化计算机使用了M7 RMOS32 (Realtime Multitasking Operating System)实时操作系统。M7 RMOS32已扩展为包含调用接口的，M7 API(Application Programming Interface，应用程序接口)已集成到SIMATIC系统中。

实时操作系统M7 RMOS32可用于对时间要求苛刻的、实时的、多任务解决方案的32位应用程序上。下列M7模块组态中可使用该操作系统：

- M7 RMOS32
- 使用MSDOS的M7 RMOS32

您为M7可编程控制系统选择的操作系统组态取决于您所使用的M7模块：

操作系统组态	模块/主存储器	PROFIBUS-DP和TCP/IP 是/否	安装在海量存储器上
M7 RMOS32	FM 356-4 / 4 MB FM 356-4 / 8 MB CPU 388-4 / 8 MB FM 456-4 / 16 MB CPU 488-3 / 16 MB CPU 486-3 / 16 MB	否 是 是 是 是 是	存储卡≥4 MB 或者硬盘
M7 RMOS32 使用MS-DOS	FM 356-4 / 8 MB CPU 388-4 / 8 MB FM 456-4 / 16 MB CPU 488-3 / 16 MB CPU 486-3 / 16 MB	否 否 是 是 是	存储卡≥4 MB 或者硬盘

26 提示和技巧

26.1 在组态表中更换模块

如果使用HW Config来修订站组态，并且，例如，希望为一个具有新订货号的站更换模块，请如下处理：

1. 使用拖放操作从硬件目录窗口将模块拖到已放置好的旧模块上。
2. 放下新模块。新模块会尽量采用已插入的模块的参数。

此步骤比通过删除旧模块、然后插入新模块并为新模块设置参数，从而更换模块的方法快。

在HW Config中，可以通过菜单命令**选项 > 设置**(“启用模块更换”)来打开或关闭此功能。

26.2 具有大量连网站的项目

如果逐个组态所有的站，然后通过菜单命令**选项 > 组态网络**调用NetPro，以便组态连接，站会自动置于网络视图中。这个步骤的缺点是，随后必须根据拓扑标准排列站和子网。

如果项目包含大量连网站，并且希望组态这些站之间的连接，应在网络视图中从头组态系统结构，以保持总览：

1. 在SIMATIC管理器中创建新项目(菜单命令**文件 > 新建**)。
2. 启动NetPro (菜单命令**选项 > 组态网络**)
3. 在NetPro中如下创建站：
   - 使用拖放操作，从目录窗口放置站。
   - 双击站以启动HW Config。
   - 在HW Config中使用拖放操作放置具有通讯能力的模块(CPU、CP、FM、IF模块)。
   - 如果希望连网这些模块，双击组态表中相应的行，创建新的子网，并连网接口。
   - 保存组态并切换到NetPro。
   - 在NetPro中，放置站和子网(用鼠标移动对象，直到到达希望的位置)
4. 在NetPro中组态连接，必要时更正连网。

26.3 重新排列

如果在STEP 7中工作时出现无法解释的问题，常常可以通过重新排列项目或库的数据库来解决。

选择菜单命令**文件 > 重新排列**，可进行重新排列。这可以清除内容删除过程中产生的间隙，这种间隙的存在会减少项目/库数据的存储空间。

这项功能可以优化项目或库的数据存储，方法类似于硬盘文件存储优化的硬盘碎片整理。

重新排列过程所需的时间取决于要移动的数据量，这可能需要一些时间。因此，该功能不能自动进行(如在关闭某个项目的时候)，而是在用户觉得需要对项目或者库进行重新排列的时候，由用户触发进行。

要求

只有当项目和库中没有任何对象被其它应用程序编辑，并因此锁定数据访问时，才能进行重新排列。

26.4 如何在多个网络上编辑符号

LAD/STL/FBD程序编辑器使您可查看和编辑多个网络的符号。

1. 点击网络名称(例如“网络1”)以选择该网络名称。
2. 按住CRTL键，并在所选网络中添加更多的网络。
3. 右击，调用关联菜单命令**编辑符号**。

使用快捷键CTRL+A，选择块中的所有网络，然后高亮度显示网络名称。

26.5 用变量表测试

为了监视和修改变量表中的变量，请注意下列编辑提示：

- 可以将符号和地址输入“符号”列以及“地址”列。条目会自动写入合适的列。
- 要显示修改的值，应将“监视”触发点设置到“扫描周期开始”处，并将“修改”触发点设置到“扫描周期结束”处。
- 如果将光标放在有红色标记的行中，将显示简要的信息，告知错误原因。按下 F1 以获得消除错误的建议。
- 只能输入已经在符号表中定义过的那些符号。
  必须完全按照符号表中的定义来输入符号。
  含特殊字符的符号名称必须包含在引号内(例如，“Motor.Off”、“Motor+Off”、“Motor-Off”)。
- 可以在“在线”标签(“自定义”对话框)中关闭警告。
- 而无需先断开连接，即可以改变连接。
- 监视触发器可以在监视变量时定义。
- 可以通过选择行并执行“强制”功能来修改所选择的变量。只修改高亮度显示的变量。
- 不确认即退出：
  如果按ESC键，当“监视”、“修改”、“释放PQ”、“监视”和“修改”终止时，不会提问是否希望退出。
- 输入连续的地址范围：
  使用菜单命令**插入 > 变量范围**。
- 显示和隐藏列：
  使用下列菜单命令显示或隐藏各个列：
  符号：**视图 > 符号**
  符号注释：**视图 > 符号注释**
  状态值的表达格式：**视图 > 显示格式**
  变量的状态值：**视图 > 状态值**
  修改变量的值：**视图 > 修改值**
- 同时改变表格多个行的显示格式：
  - 按住鼠标左键，在目标表格区域上拖动，选择希望改变显示格式的区域。
  - 用菜单命令**视图 > 选择显示格式**选择表达方式。仅改变那些允许改变格式的、选中的表格行的格式。
- 通过 F1 键输入实例：
  - 如果将光标放在地址列并按 F1，将获得地址输入的实例。
  - 如果将光标放在修改值列并按 F1，将获得修改/强制输入值的实例。

26.6 使用程序编辑器修改变量

在程序编辑器中，您可对用于二进制输入和存储位的按钮进行编程，这些按钮将为您提供一种快速、简单的方式，通过鼠标点击对这些地址进行修改。

要求

- 在符号表中，通过菜单命令**特定对象属性 > 接触控制**，您已将该属性分配给您想要修改的地址
- 已经选择了LAD/STL/FBD程序编辑器“常规”标签中的“接触控制”选项(菜单命令**选项 > 自定义**)。
- 已经选择了菜单命令**调试 > 监视**。

此处的触发条件为“永久/周期启动处”。

只要持续按下按钮，就会对设备实际使用的输入进行监视。您还可以通过多重选择(CTRL键)修改多个输入。

至于位存储器或无法使用的输入，按下按钮将使状态置位为1。仅当通过关联菜单条目或在变量表中提出明确要求，或地址被STEP 7程序复位时，该状态才会复位为0。

至于非负输入或位存储器，按下按钮将导致修改值“1”生效；至于负的输入或位存储器，则修改值“0”生效。

WinCC注意事项

如果通过操作员控制和变量监视在WinCC中启动了程序编辑器，只允许使用WinCC的控制选项。否则，如果操作员已经获得WinCC的“维护权”，则两个修改选项都允许使用。

26.7 虚拟工作存储器

STEP 7 中出现问题的另一个原因可能是虚拟工作存储器不够。

使用 STEP 7 进行工作时，您应该调整虚拟存储器设置。操作过程如下：

1. 例如，打开控制面板，进入开始菜单**开始 > 设置 > 控制面板**，再双击“系统”图标。
   仅适用于XP：在**开始 > 桌面 > 属性 > 高级 > 系统性能 > 设置**下打开。
2. 在Windows 2000中，选择“高级”标签，并点击“系统性能选项”按钮。
   在Windows XP中，在“系统设置”对话框中选择“高级”标签。
3. 点击“更改”按钮。
4. 在“最小值”中输入至少40兆字节，在“最大值”中输入至少150兆字节。

注意

由于虚拟存储器位于硬盘上(缺省为C:)并且是动态的，请确保TMP或TEMP目录有足够的存储器(约20 - 30兆字节)：

- 如果S7项目与虚拟存储器在同一个分区上，则应留有两倍于S7项目的空闲存储空间。
- 如果项目存储在另一个分区上，则没有这种要求。

A 附录

A.1 工作模式

A.1.1 工作模式和模式转换

工作模式

工作模式描述了特定时间点处 CPU 的行为。了解 CPU 的工作模式对编程启动、测试控制器以及进行故障诊断非常有用。

S7-300 和 S7-400 CPU 可采用下列工作模式：

- STOP
- STARTUP
- RUN
- HOLD

在 STOP 模式中，CPU 会检查所有已组态的模块或由缺省寻址设置的模块是否的确存在，并将 I/O 设置为预定义的初始状态。用户程序不在 STOP 模式下执行。

在 STARTUP 模式中，“暖启动”、“冷启动”和“热启动”启动类型之间互有区别：

- 在暖启动中，从程序开始处以系统数据和用户地址区的初始设置开始进行程序处理(非保持性定时器、计数器和位存储器复位)。
- 在冷启动中，执行 OB1 中的第一个命令时，读取过程映像输入表，并处理 STEP 7 用户程序(也适用于暖启动)。
  - 删除工作存储器中由 SFC 创建的数据块；剩余的数据块具有来自装入存储器的预置值。
  - 复位过程映像和所有定时器、计数器和位存储器，不管它们是否具有保持性分配。
- 在热启动中，在程序中断处重新开始执行程序(不复位定时器、计数器和位存储器)。只有 S7-400 CPU 中才能进行热启动。

在 RUN 模式中，CPU 执行用户程序、更新输入和输出、处理中断并处理过程错误消息。

在 HOLD 模式中，暂停用户程序处理，然后可以逐步测试用户程序。只有在使用编程设备进行测试时才能使用 HOLD 模式。

在所有这些模式中，CPU 可通过多点接口(MPI)进行通讯。

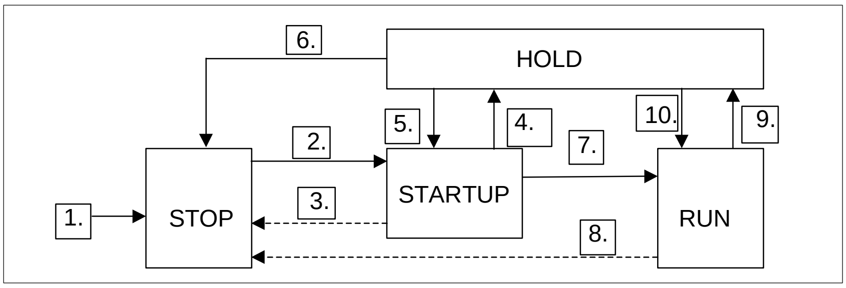
其它工作模式

如果 CPU 没有准备好操作，那么它处于下列其中一种模式中：

- 关闭，即切断电源。
- 故障，即发生故障。
  要检查 CPU 是否的确发生故障，可将 CPU 切换到 STOP 模式，然后切断电源，再接通电源。如果 CPU 启动，则显示诊断缓冲区，以便分析问题。如果 CPU 没有启动，则必须替换该 CPU。

工作模式转换

下图显示了 S7-300 和 S7-400 CPU 的工作模式和模式转换：



下表显示了可改变工作模式的条件。

转换	描述
1.	接通电源后，CPU 处于 STOP 模式。
2.	CPU 进入 STARTUP 模式： • 通过钥匙开关或编程设备将 CPU 转变为 RUN 或 RUN-P 之后。 • 通过接通电源自动触发启动后。 • 执行了 RESUME 或 START 通讯功能。 在两种情况下，钥匙开关必须设置为 RUN 或 RUN-P。
3.	在下列情况下，CPU 返回 STOP 模式： • 启动期间检测到错误。 • 通过钥匙开关或编程设备，将 CPU 变为 STOP 模式。 • 在启动 OB 中执行了停止命令。 • 执行了 STOP 通讯功能。
4.	到达启动程序中的断点时，CPU 变为 HOLD 模式。
5.	当启动程序中的断点置位，且“EXIT HOLD”命令执行后，CPU 变为 STARTUP 模式(测试功能)。

转换	描述
6.	在下列情况下，CPU 返回 STOP 模式： • 通过钥匙开关或编程设备，将 CPU 变为 STOP 模式。 • 执行了 STOP 通讯命令。
7.	如果启动成功，那么 CPU 进入 RUN 模式。
8.	在下列情况下，CPU 返回 STOP 模式： • 在 RUN 模式中检测到错误，并且没有装载相应的 OB。 • 通过钥匙开关或编程设备，将 CPU 变为 STOP 模式。 • 在用户程序中编辑停止命令。 • 执行了 STOP 通讯功能。
9.	到达用户程序中的断点时，CPU 变为 HOLD 模式。
10.	置位一个断点并执行“EXIT HOLD”命令之后，CPU 进入 RUN 模式。

工作模式优先级

如果同时请求大量工作模式转换，那么选择具有最高优先级的工作模式。例如，如果模式选择器设为 RUN，而用户尝试在编程设备中将 CPU 设为 STOP，那么由于该模式优先级最高，因此 CPU 将进入 STOP 模式。

优先级	模式
最高	STOP
	HOLD
	STARTUP
最低	RUN

A.1.2 STOP 模式

用户程序不能在 STOP 模式下执行。所有输出都被设置为替换值，以便使受控过程处于安全状态。CPU 进行以下检查：

- 硬件是否有问题(例如：模块不可用)？
- CPU 应采用缺省设置，或者进行参数设置？
- 是否满足程序中规定的启动条件？
- 有没有系统软件问题？

在 STOP 模式下，CPU 也能接收全局数据，且有可能为已组态的连接使用通讯 SFB，为未组态的连接使用通讯 SFC 进行被动单向通讯。

存储器复位

在 STOP 模式下，CPU 存储器可以复位。可以通过按键开关(MRES)对存储器进行手动复位，或者通过编程设备进行复位(如在下载某个用户程序之前)。

CPU 存储器复位意味着 CPU 将回到初始状态，如下所示：

- 工作存储器和 RAM 装载存储器中全部的用户程序，以及所有地址区域，都被清零。
- 系统参数以及 CPU 和模块参数复位为缺省设置。在存储器复位前设置的 MPI 参数保留。
- 如果插入了存储卡(EPROM 闪存)，CPU 就会将存储卡中的用户程序复制到工作存储器中(如果存储卡中有合适的组态数据，则还包括 CPU 和模块参数)。

诊断缓冲区、MPI 参数、时间以及运行系统计时器不会复位。

A.1.3 STARTUP 模式

在 CPU 可以开始处理用户程序之前，必须首先执行启动程序。通过在启动程序中编程启动 OB，可以指定循环程序的某些特定设置。

有三种类型的启动：暖启动、冷启动和热启动。只有 S7-400 CPU 中才能进行热启动。必须通过 STEP 7 在 CPU 的参数集中显式设置这一点。

STARTUP 模式的特性如下：

- 处理启动 OB 中的程序(OB100 用于暖启动，OB101 用于热启动，OB102 用于冷启动)。
- 不能执行时间驱动型或中断驱动型程序。
- 更新定时器。
- 运行时测量器开始运行。
- 信号模块上禁止的数字量输入(可通过直接访问设置)。

暖启动

始终允许暖启动，除非系统已经请求存储器复位。发生下列情况后，暖启动是唯一选项：

- 存储器复位
- 当 CPU 处于 STOP 模式时，下载用户程序
- I 栈/B 栈溢出
- 放弃暖启动(由于断电或改变模式选择器设置)
- 当热启动前的中断超过选中的时间限制时。

手动暖启动

手动暖启动可由下列各项触发：

- 模式选择器

  (CRST/WRST 开关 - 如果可用 - 必须设置为 CRST)
- 编程设备上的相应命令或通过通讯功能

  (如果模式选择器设为 RUN 或 RUN-P)

自动暖启动

在下列情况下，上电后可触发自动暖启动：

- 发生断电时，CPU 不处于 STOP 模式。
- 模式选择器设为 RUN 或 RUN-P。
- 没有编程上电后自动热启动。

- 在暖启动期间，CPU 因断电中断(与编程设定的重启动类型无关)。

CRST/WRST 开关对自动暖启动没有影响。

无备用电池时自动暖启动

如果使用没有备用电池的 CPU(如果有必要执行免维护操作)，那么在接通电源后或断电恢复电源后，CPU 存储器自动复位，并执行暖启动。用户程序必须位于闪存 EPROM(存储卡)上。

热启动

在 RUN 模式下断电、恢复电源后，S7-400CPU 执行一个初始化例行程序，然后自动执行热启动。在热启动期间，用户程序从中断处重新执行。断电之前未执行的用户程序段被称为剩余周期。剩余周期可包含时间驱动型和中断驱动型程序段。

仅当用户程序没有在 STOP 模式中修改(例如，重新装载一个已修改的块)，且没有其它暖启动原因时，才允许热启动。手动和自动执行热启动均可。

手动热启动

仅当 CPU 参数集中有合适的参数设置并由于下列原因导致 STOP 时，才能使用手动热启动：

- 模式选择器从 RUN 变为 STOP。
- 没有装载用户编程的 STOP、调用 OB 后的 STOP。
- STOP 模式是编程设备上一个命令或通讯功能的结果。

可由下列各项触发手动热启动：

- 模式选择器

  CRST/WRST 必须设置为 WRST。
- 编程设备上的相应命令或通过通讯功能(模式选择器设为 RUN 或 RUN-P)。
- 在 CPU 的参数集中设置手动热启动时。

自动热启动

在下列情况下，上电后可触发自动热启动：

- 发生断电时，CPU 不处于 STOP 或 HOLD 模式。
- 模式选择器设为 RUN 或 RUN-P。
- 在 CPU 的参数集中设置上电后自动热启动。

CRST/WRST 开关对自动热启动没有影响。

断电后的保持性数据区

S7-300 和 S7-400 CPU 对断电再上电的响应不同。

S7-300 CPU (除了 CPU318 外)只具有暖启动的能力。然而，通过 STEP 7，可以将存储位、定时器、计数器以及数据块中的区指定为具有保持性，以免由于断电而造成数据丢失。当上电时，执行存储器自动暖启动。

S7-400 CPU 根据参数设置，以暖启动(保持性或非保持性上电后)或热启动(只能在保持性上电后)响应重新上电。

下表显示了在暖启动、冷启动或热启动期间，S7-300 和 S7-400 上保持的数据。

X	表示	保持的数据
VC	表示	在 EPROM 上保持的逻辑块，所有过载的逻辑块均丢失
VX	表示	只有当 EPROM 上的保持数据来自 NV-RAM 时，才保持数据块(RAM 中装载或创建的数据块丢失)
0	表示	复位或删除数据(DB 内容)
V	表示	数据设置为从 EPROM 存储器中获取的初始值
---	表示	不可能，因为 NV-RAM 不可用

下表显示了在工作存储器(EPROM 和 RAM 装入存储器)中保持的数据：

			EPROM	**(存储器**	**卡**	**或**	**集成)**		
	CPU	**带**	**备份**	**电池**		**CPU**	**而无需**	**备份**	**电池**
数据	装入存储器中的块	工作存储器中的 DB	存储位、定时器、计数器	存储位、定时器、计数器	装入存储器中的块	工作存储器中的 DB	工作存储器中的 DB	存储位、定时器、计数器	存储位、定时器、计数器
			(定义为保持性)	(定义为易失性)		(定义为保持性)	(定义为易失性)	(定义为保持性)	(定义为易失性)
S7-300 上的暖启动	X	X	X	0	VC	VX	V	X	0
S7-400 上的暖启动	X	X	X	0	VC	---	V	0	0
S7-300 上的冷启动 S7-300	X	0	0	0	VC	V	V	0	0
S7-300 上的冷启动 S7-400	X	0	0	0	VC	---	V	0	0
S7-400 上的热启动	X	X	X	X		只能	进行	暖启动	

启动活动

下表显示启动期间，CPU 执行的活动：

以执行顺序排列的活动	在暖启动中	在冷启动中	在热启动中
清除 I 栈/B 栈	X	X	0
清除易失性存储位、定时器、计数器	X	0	0
清除所有存储位、定时器、计数器	0	X	0
清除过程映像输出表	X	X	可选
复位数字量信号模块的输出	X	X	可选
放弃硬件中断	X	X	0
放弃延时中断	x	x	0
放弃诊断中断	X	X	X

以执行顺序排列的活动	在暖启动中	在冷启动中	在热启动中
更新系统状态列表(SZL)	X	X	X
计算模块参数并将其传送到模块或传送缺省值	X	X	X
执行相关启动 OB	X	X	X
执行剩余周期(由于断电没有执行的程序部分)	0	0	X
更新过程映像输入表	X	X	X
转换到 RUN 后允许数字量输出(取消 OD 信号)	X	X	X
X　表示　已执行 0　表示　未执行			

放弃启动

如果在启动期间发生错误，那么放弃启动，CPU 进入或保持 STOP 模式。

必须重复放弃的暖启动。放弃重启动后，可进行暖启动和热启动。

在下列情况下，不执行启动(重启(暖启动)或热启动)或放弃启动：

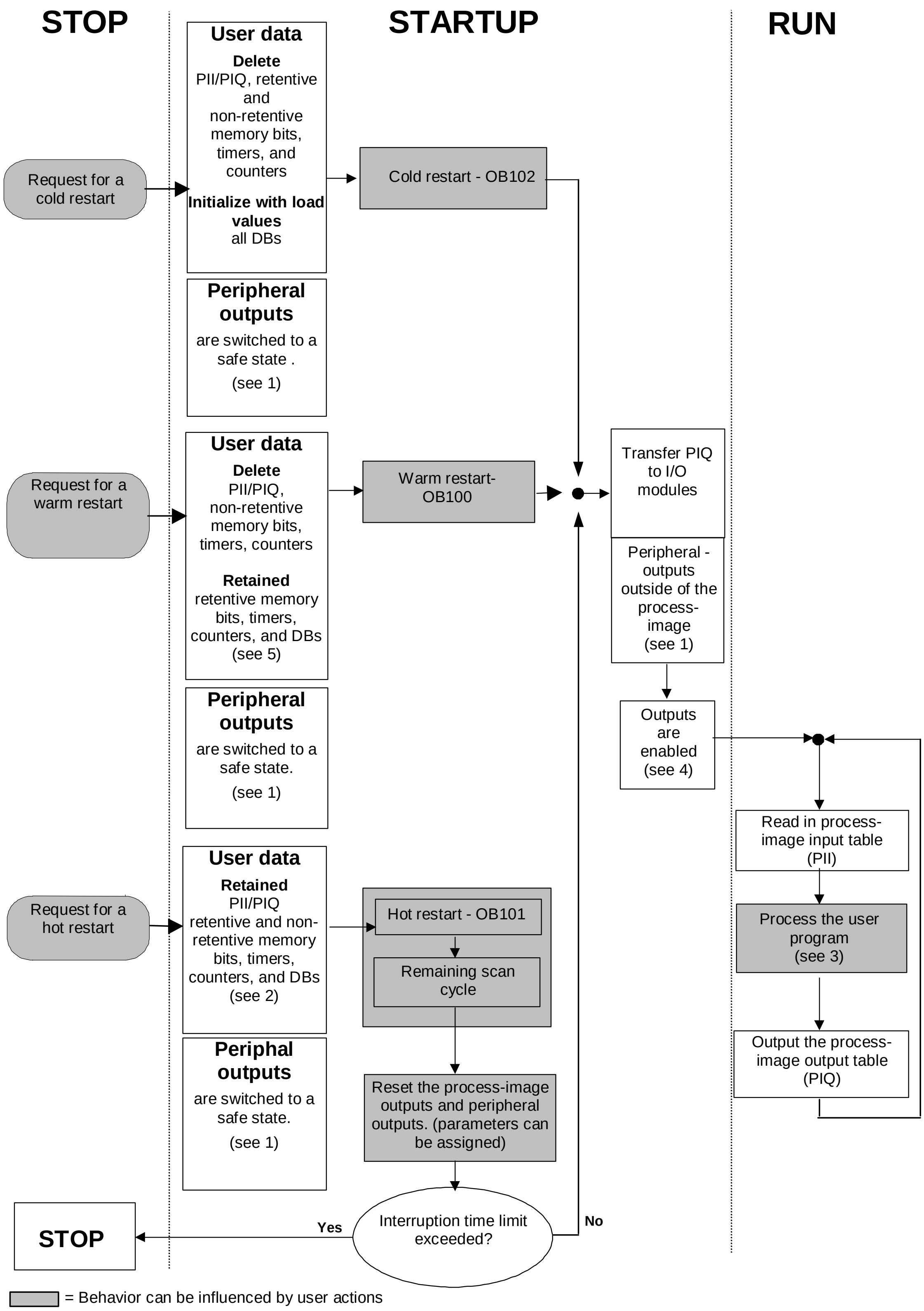
- CPU 的工作模式开关设为 STOP。
- 请求存储器复位。
- 插入带 STEP 7 不许可应用代码的存储卡(例如，STEP 5)。
- 在单个处理器模式中插入一个以上 CPU。
- 如果用户程序包含 CPU 不能识别或已经禁止的 OB。
- 如果在上电后，CPU 发现通过 STEP 7 创建的组态表中列出的所有模块实际上没有全部插入(不允许预置和实际参数分配之间出现区别)。
- 如果在计算模块参数时发生错误。

在下列情况下，不执行热启动或放弃热启动：

- CPU 存储器复位(存储器复位后只能执行暖启动)。
- 超过中断时间限制(该时间为从退出 RUN 模式到执行完包含剩余周期的启动 OB 之间的时间)。
- 改变了模块组态(例如，替换了模块)。
- 参数分配只允许暖启动。
- 当 CPU 处于 STOP 模式时，装载、删除或修改了块时。

活动顺序

下图显示了在 STARTUP 和 RUN 期间 CPU 的活动：



“STARTUP 和 RUN 期间 CPU 的活动”图的要点

1. 通过 I/O 模块在硬件侧将所有外围设备输出都切换到安全状态(缺省值 = 0)。不管用户程序采用的是过程映像区以内的输出，还是以外的输出，都可进行该切换。

   如果使用具有替换值能力的信号模块，那么可以将参数分配给输出特性，如“保持上一次值”。

2. 处理剩余扫描周期时必需。
3. 在首次调用中断 OB 时，该 OB 可使用当前过程映像输入表。
4. 通过下列步骤，可在用户程序的第一个扫描周期内确定本地和分布式外围设备输出的状态：
   - 使用可给其分配参数的输出模块来启用替换值输出或保持上一次值。
   - 对于热启动：激活“在热启动期间复位输出”CPU 启动参数，以输出 0(相当于缺省设置)。
   - 在启动 OB(OB100、OB101、OB102)中预置输出。
5. 在没有备份的 S7-300 系统中，只有组态为保持性的 DB 区才能保持。

A.1.4 RUN 模式

在 RUN 模式下，CPU 执行周期程序、时间驱动程序和中断驱动程序，如：

- 读取输入的过程映像。
- 执行用户程序。
- 输出过程映像输出表。

只有在 RUN 模式下，才有可能通过全局数据通讯(全局数据表)，且为已组态的连接使用通讯 SFB，为未组态的连接使用通讯 SFC，进行 CPU 之间的主动数据交换。.

下表给出了不同工作模式下，有可能进行的数据交换的实例：

通讯类型	**CPU 1 的模式**	**数据交换方向**	**CPU 2 的模式**
全局数据通讯	RUN	↔	RUN
	RUN	→	STOP/HOLD
	STOP	←	RUN
	STOP	X	STOP
	HOLD	X	STOP/HOLD
单向通讯	RUN	→	RUN
带有通讯 SFB	RUN	→	STOP/HOLD

通讯类型	CPU 1 的模式	数据交换方向	CPU 2 的模式
双向通讯，带有通讯 SFB	RUN	↔	RUN
单向通讯	RUN	→	RUN
带有通讯 SFC	RUN	→	STOP/HOLD
双向通讯，带有通讯 SFC	RUN	↔	RUN
↔ 表示 数据交换可以双向进行 → 表示 数据交换只能单向进行 X 表示 不能进行数据交换			

A.1.5 HOLD 模式

HOLD 模式是一种特殊的模式。仅用于在启动阶段或者 RUN 模式下的测试目的。HOLD 模式下会出现下列情况：

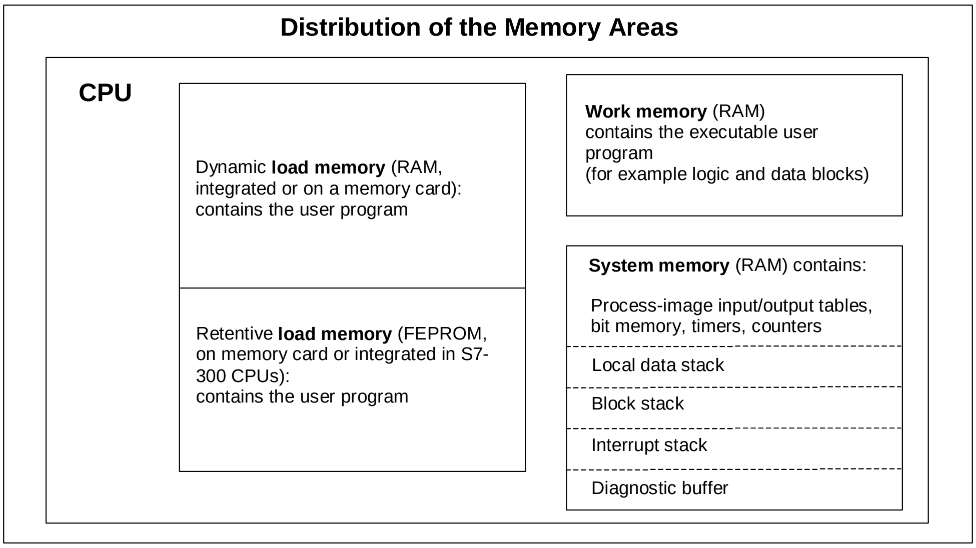
- 所有的定时器被冻结：定时器和运行系统计时器不工作，监视时间停止，时间驱动电平的基本时钟脉冲停止。
- 实时时钟运行。
- 输出未激活，但可以为测试的目进行激活。
- 可以置位和复位输入和输出。
- 在 HOLD 模式下，当有备用电池的 CPU 掉电，恢复供电后，CPU 变为停止状况，而不是自动热启动或重启动(暖启动)。恢复供电后，没有备用电池的 CPU 将自动重启动(暖启动)。
- 可以接收全局数据，且有可能为组态的连结使用通讯 SFB，为非组态的连结使用通讯 SFC，进行被动单向通讯(参见 RUN 模式节中的表)。

A.2 S7 CPU 的存储区

A.2.1 存储器区的分配

S7 CPU 的存储器可以划分为三个区域(参见下图)：

- 装入存储器用于没有符号地址分配或注释的用户程序(这些保留在编程设备的存储器中)。装入存储器可以是 RAM 或 EPROM。
- 未标记为启动时所需要的块将只存储在装入存储器中。
- 工作存储器(集成的 RAM)包含了与运行程序相关的部分 S7 程序。该程序仅在工作存储器和系统存储器区中执行。
- 系统存储器(RAM)包含了每个 CPU 为用户程序提供的存储器单元，例如过程映像输入和输出表、位存储器、定时器和计数器。系统存储器也包含块堆栈和中断堆栈。
- 除了上述的区域外，CPU 的系统存储器还提供了临时存储器(局部数据堆栈)，存放调用块时用到的临时数据。这些数据只在块激活时才保持有效。



A.2.2 装入存储器和工作存储器

当从编程设备下载用户程序到 CPU 时，只有逻辑和数据块被装载到 CPU 的装入存储器和工作存储器中。

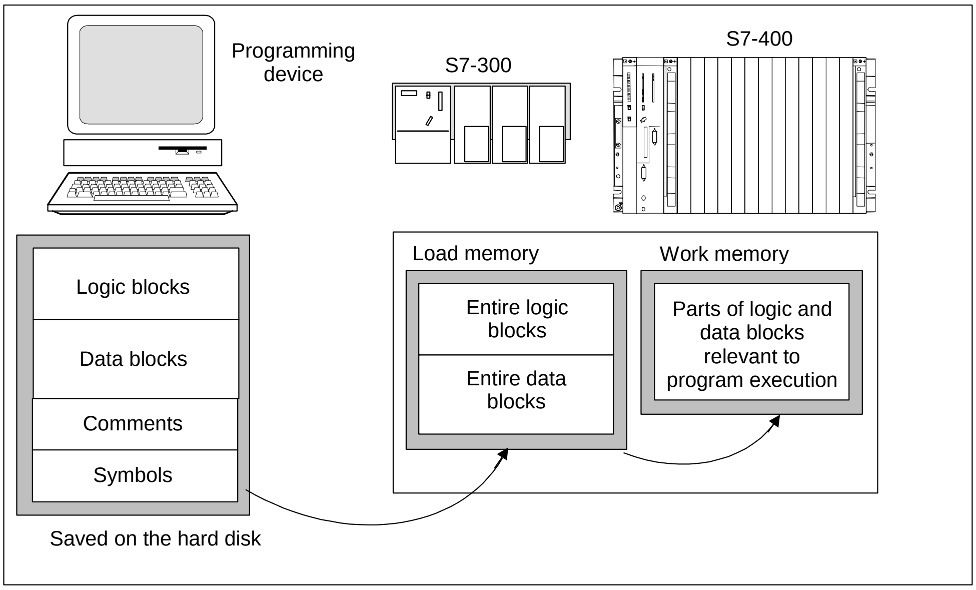
符号地址分配(符号表)和块注释保留在编程设备上。

分割用户程序

为保证快速执行用户程序，以及避免到不能扩展的工作存储器的不必要的的装载，只装载与程序执行相关的部分到工作存储器。

执行程序时不需要的块的各部分(例如，块标题)保留在装入存储器中。

下图给出了一个被装载到 CPU 存储器中的程序。



注意

借助系统功能，在用户程序中创建的数据块(例如 SFC22 CREAT_DB)被 CPU 整个地保存在工作存储器中。

某些 CPU 在工作存储器中有用于代码和数据的独立管理区域。这些区域的大小和分配在这些 CPU 的模块信息的“存储器”标签中给出。

识别数据块为“与执行无关”

在源文件中编程的数据块，作为 STL 程序的一部分，就可以被识别为“与执行无关”(关键字 UNLINKED)。这意味着当它们下载到 CPU 时，数据块只存储在装入存储器中。如有必要，可以使用 SFC20 BLKMOV 将这些块的内容复制到工作存储器。

这项技术节省了工作存储器的空间。因此可扩展装入存储器用作缓存区(例如，对用于混和的公式：只有用于下一批的公式才装载到工作存储器中)。

装入存储器结构

装入存储器可以用存储卡扩展。为获得装入存储器的最大容量，参见“S7-300 可编程控制器、硬件及安装手册”和“S7-400、M7-400 可编程控制器模块技术规范参考手册”装入存储器。

在 S7-300 CPU 中装入存储器也可以具有集成的 EPROM 部分，以及集成的 RAM 部分。通过在 STEP 7 中分配参数，数据块中的区域可以声明为保持(参见 S7-300 CPU 上保持存储器区)。

在 S7-400 CPU 中，使用存储卡(RAM 或 EEPROM)以扩展装入存储器是强制性的。集成的装入存储器是 RAM 存储器，主要用于重新装载和更正块。对于新的 S7-400 CPU，也可以插入附加的工作存储器。

RAM 和 EPROM 区域中的装入存储器特性

根据是选择 RAM 还是 EPROM 存储卡来扩展装入存储器，装入存储器在下载、重新装载或存储器复位期间可能有不同的反应。

下表给出了多种装载方法：

存储器类型	装载方法	装载类型
RAM	下载并删除各个块	PG-CPU 连接
	下载并删除整个 S7 程序	PG-CPU 连接
	重新装入各个块	PG-CPU 连接
集成的(仅适用于 S7-300)或插入式 EPROM	下载整个 S7 程序	PG-CPU 连接
插入式 EPROM	下载整个 S7 程序	上传 EPROM 到 PG，并将存储卡插入 CPU 下载 EPROM 到 CPU

当复位 CPU 存储器(MRES)或拆除 CPU 或 RAM 存储卡时，存储在 RAM 中的程序将丢失。

保存在 EPROM 存储卡中的程序不会因 CPU 存储器复位而被擦除，甚至在没有备用电池的情况下，也会保留(传输、备份副本)。

A.2.3 系统存储器

A.2.3.1 使用系统内存区域

S7 CPU 的系统存储器被划分成多个地址区(参见下表)。使用程序中的指令，可以在相应的地址区域中直接对数据寻址。

地址区	通过下列范围的单元进行访问	**S7 符号(IEC)**	描述
过程映像输入表	输入(位)	I	在扫描周期的开始，CPU 从输入模块读取输入，并记录该区域中的值。
	输入字节	IB	
	输入字	IW	
	输入双字	ID	
过程映像输出表	输出(位)	Q	在扫描周期期间，程序计算输出值并将它们放入此区域。在扫描周期结束时，CPU 发送计算的输出值到输出模块。
	输出字节	QB	
	输出字	QW	
	输出双字	QD	
位存储器	存储器(位)	M	此区域用于存储程序中计算的中间结果。
	存储器字节	MB	
	存储器字	MW	
	存储器双字	MD	
定时器	定时器(T)	T	此区域为定时器提供存储空间。
计数器	计数器(C)	C	此区域为计数器提供存储空间。
数据块	数据块，用“OPN DB”打开：	DB	数据块包含程序的信息。它们可以被由所有逻辑块定义为通用(共享 DB)，或者可以分配给特定的 FB 或 SFB (背景数据块)。
	数据位	DBX	
	数据字节	DBB	
	数据字	DBW	
	数据双字	DBD	
	数据块，用“OPN DI”打开：	DI	
	数据位	DIX	
	数据字节	DIB	
	数据字	DIW	
	数据双字	DID	

地址区	通过下列范围的单元进行访问	S7 符号(IEC)	描述
本地数据	本地的数据位	L	当块被执行时，此区域包含块的临时数据。L 堆栈也提供存储空间，用于传送块参数和记录来自梯形图程序段的中间结果。
	本地的数据字节	LB	
	本地的数据字	LW	
	本地的数据双字	LD	
外设(I/O)区： 输入	外设输入字节	PIB	外围设备输入和输出区域允许直接访问中央和分布式的输入和输出模块(DP)。
	外设输入单字	PIW	
	外设输入双字	PID	
外设(I/O)区： 输出	外设输出字节	PQB	
	外设输入双字	PQW	
	外设输出双字	PQD	

参见下列 CPU 手册或指令列表，以获取关于哪个地址区域可用于您的 CPU 的信息：

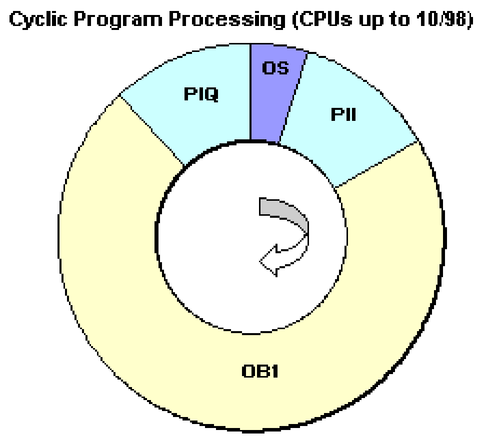
- “S7-300 可编程控制器，硬件与安装”手册
- “S7-400、M7-400 可编程控制器，模块技术规范”参考手册
- “S7-300 可编程控制器，指令列表”
- “S7-400 可编程控制器，参考指南”

A.2.3.2 过程映像输入/输出表

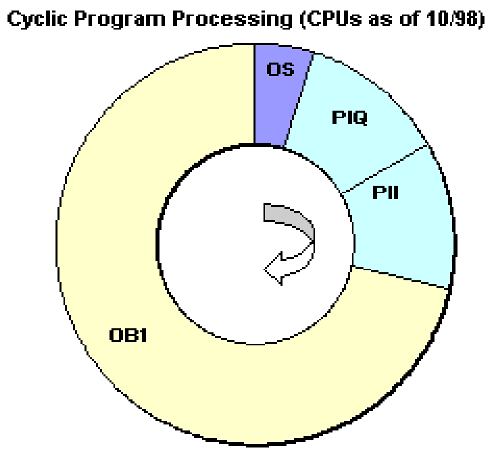
如果输入(I)和输出(Q)地址区在用户程序中访问，程序并不扫描数字信号模块上的信号状态，而是访问 CPU 系统存储器和分布式 I/O 中的存储器区。该存储器区就是过程映像。

更新过程映像

下图给出在一个扫描周期内的处理步骤。



操作系统(OS)的一个内部任务将读取输入状态到过程映像输入表(PII)。一旦该步骤完成，将执行用户程序以及它所调用的所有块。周期结束时，将过程映像输出表(PIQ)写入到模块的输出。读入过程映像输入表并将过程映像输出表写入模块的输出，均由操作系统独立控制。



操作系统(OS)的一个内部任务将过程映像输出表(PIQ)写入模块的输出，并读取输入状态到过程映像输入表(PII)。一旦该步骤完成，将执行用户程序以及它所调用的所有块。将过程映像输出表写入模块的输出和读入过程映像输入表均由操作系统独立控制。

过程映像的优势

与直接访问输入/输出模块相比，过程映像访问的主要优势在于在一个程序周期持续期间，CPU 具有过程信号的一致性的映像。如果在程序执行期间，输入模块的信号状态发生了变化时，过程映像中的信号状态仍被保持，直到下一个周期过程映像进行了更新。在用户程序中周期性地扫描输入信号的过程，确保了总有一致的输入信息。

访问过程映像还比直接访问信号模块更节省时间，因为过程映像位于 CPU 的存储器中。

局部过程映像(过程映像分区)

除了由操作系统自动更新的过程映像(过程映像输入表 PII 和过程映像输出表 PIQ)，还可为 S7-400 CPU 分配最多 15 个局部过程映像(CPU 专用的，no. 1 到 no. 15，参见 S7-400、M7-400 可编程控制器模块技术规格参考手册)。也就是说，在必要时，可以独立于过程映像表的更新周期，更新过程映像表的部分。

通过 STEP 7 为每个输入/输出地址分配的过程映像分区将不再属于 OB1 过程映像输入/输出表。输入和输出地址只能一次分配到 OB 1 过程映像和所有的过程映像分区。

在分配地址时，可以使用 STEP 7 定义过程映像分区(哪些模块输入/输出地址列在哪些过程映像分区)。过程映像分区既可由用户通过 SFC 来更新，也可通过系统连接的 OB 进行自动更新。

例外：同步周期中断 OB 的过程映像分区并不在系统侧更新，即使它们已链接到 OB (OB 61 到 OB 64)上。

注意

对于 S7-300 CPU，未分配的过程映像输入和输出可以用作附加的位存储区域域。使用该性能的程序只要满足下列条件之一，就可运行在更低版本的(即，4/99 之前的) S7-400 CPU 上：

对于这些 S7-400 CPU

- 用作位存储器的过程映像区必须位于参数赋值“过程映像的大小”以外，或者
- 必须位于既不被系统也不被 SFC26/SFC27 更新的过程映像分区。

使用 SFC 更新局部过程映像(过程映像分区)

可以在用户程序中使用 SFC 来更新整个过程映像或一个过程映像分区。

- 要求：正在处理中的过程映像不能由系统更新。
- SFC26 UPDAT_PI：更新过程映像输入表
- SFC27 UPDAT_PO：更新过程映像输出表。

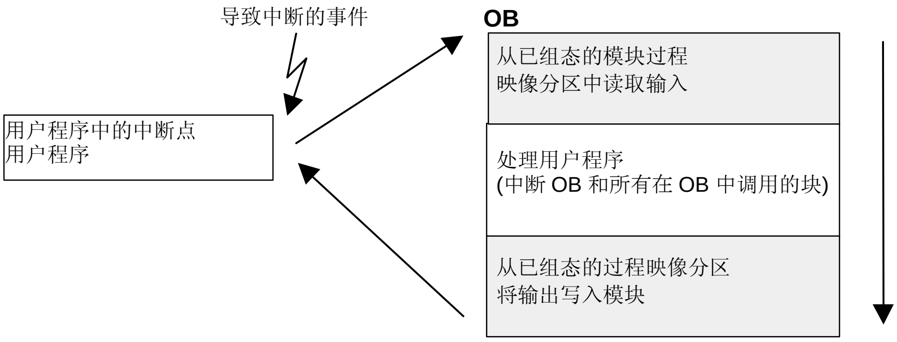
局部过程映像(过程映像分区)的系统更新

也可以调用 OB 自动对过程映像分区进行系统更新 - 与(整个)过程映像类似，它是在 OB1 处理之前或之后循环地更新。只能作为一个参数为特定的 CPU 分配该功能。

在运行期间，所分配的过程映像分区将自动更新：

- 在处理 OB 之前，用于输入的过程映像分区
- 在处理 OB 之后，用于输出的过程映像分区

在分配 OB 优先级的同时，可以为 CPU 分配参数，来指示哪个过程映像分区被分配给了哪个 OB。



在过程映像更新期间的 I/O 访问错误(PZF)

在过程映像更新时，CPU 系列缺省情况下(S7-300 和 S7-400)对错误的反应有差别：

- S7-300：不在诊断缓冲区中生成条目，不调用 OB，相应的输入字节被复位为“0”并将保持为“0”，直到故障消失。
- S7-400：在诊断缓冲区中生成一个条目，为相应的每个过程映像更新的每个 I/O 访问启动 OB85。每次访问过程映像时，故障输入字节被复位为“0”。

对于新型的 CPU (如 4/99)，可以为 I/O 访问错误的反应重新分配参数，以便 CPU 以下列方式之一工作：

- 在诊断缓冲区中生成条目，仅为进入的和离开的 PZF 启动 OB85(在调用 OB 85 之前，故障输入字节已复位为“0”，而且在 PZF 离开之前操作系统不会再将其覆盖)
- 生成缺省的 S7-300 执行结果(不调用 OB85；相应的输入字节被复位为“0”，并且故障清除之前操作系统不会再将其覆盖。)
- 生成缺省的 S7-400 执行结果(为每个单独访问调用 OB85；每次访问过程映像时将故障输入字节复位为“0”。)

OB85 的启动频率？

除了作为参数分配的 PZF 的反应(进入的/离开的或对于每个 I/O 访问)之外，模块的地址空间也会影响 OB85 的启动频率：

对于一个地址空间多达双字的模块，OB85 启动一次，例如对于最多为 32 位的输入或输出的数字模块或对于有两个通道的模拟模块。

对于有更大的地址空间的模块，OB85 的启动次数随双字命令需要的访问次数而定，例如，对于带有四个通道的模拟模块需要两次。

A.2.3.3 局部数据堆栈

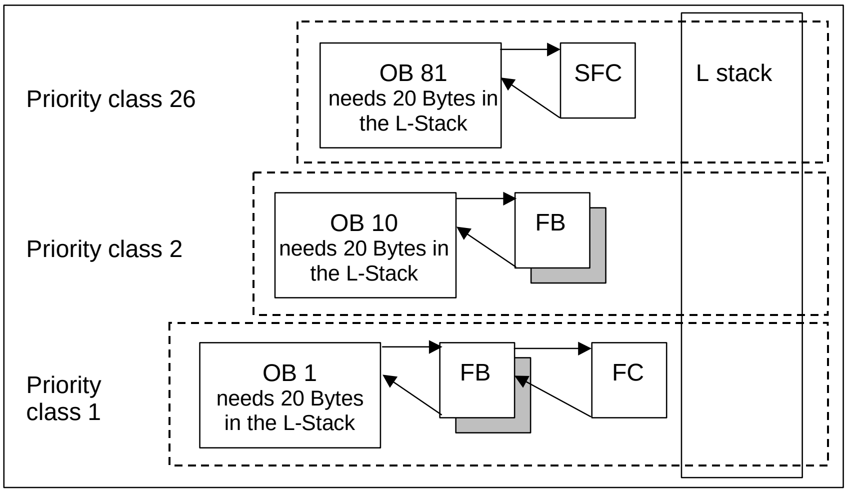
L 堆栈保存有：

- 块的局部数据的临时变量
- 组织块的启动信息
- 关于传送参数的信息
- 梯形图程序中逻辑的中间结果

当您对组织块编程时，可以声名临时变量(TEMP)只在块执行期间可用，然后它将被覆盖。在首次访问局部数据堆栈之前，必须对局部数据初始化。除此之外，每个组织块还需要 20 个字节的局部数据来存储它们的启动信息。

CPU 只能为当前执行的块的临时变量(局部数据)提供有限的存储空间。该存储器区局部数据堆栈的大小取决于 CPU。局部数据堆栈被各优先级均分(缺省)。也就是说每个优先级都有它自己的局部数据区，从而保证了较高的优先级和它们的 OB 自身的局部数据有可用的空间。

下图用一个实例给出了优先级的局部数据分配，其中在 L 堆栈中，OB1 被 OB10 中断，而后者又被 OB81 中断。



警告

OB 和它相关的块中所有的临时变量(TEMP)都保存在 L 堆栈中。如果使用了过多的嵌套的层，在执行块时，L 堆栈可能会溢出。

如果一个允许的 L 堆栈大小超出，CPU 将切换到 STOP 模式。

测试程序中的 L 堆栈(临时变量)。

同步错误 OB 的局部数据需求也必须加以考虑。

分配局部数据给优先级

局部数据堆栈中，并不是每个优先级都需要同样大小的存储空间。在 STEP 7 中，通过分配参数，可以为 S7-400 CPU 和 CPU 318 的各个优先级分配不同大小的局部数据区。任何不需要的优先级都可以将其取消。这样 S7-400 CPU 和 CPU 318 中其它优先级等级的存储器区便增大了。在程序执行期间，取消激活的 OB 将被忽略，以节省周期。

而对于其它的 S7-300 CPU，每个优先级被分配了固定大小的局部数据(256 个字节)，不能对其修改。

A.2.3.4 中断堆栈

如果程序的执行被更高优先级的 OB 中断，操作系统将保存累加器和地址寄存器的当前内容，以及在中断堆栈中打开数据块的编号和长度。

一旦执行新的 OB，操作系统将从 I 堆栈中装载信息，并从中断发生处恢复执行中断的块。

当 CPU 处于 STOP 模式时，可以使用 STEP 7 在编程设备上显示 I 堆栈。这就允许您找出 CPU 改变为 STOP 模式的原因。

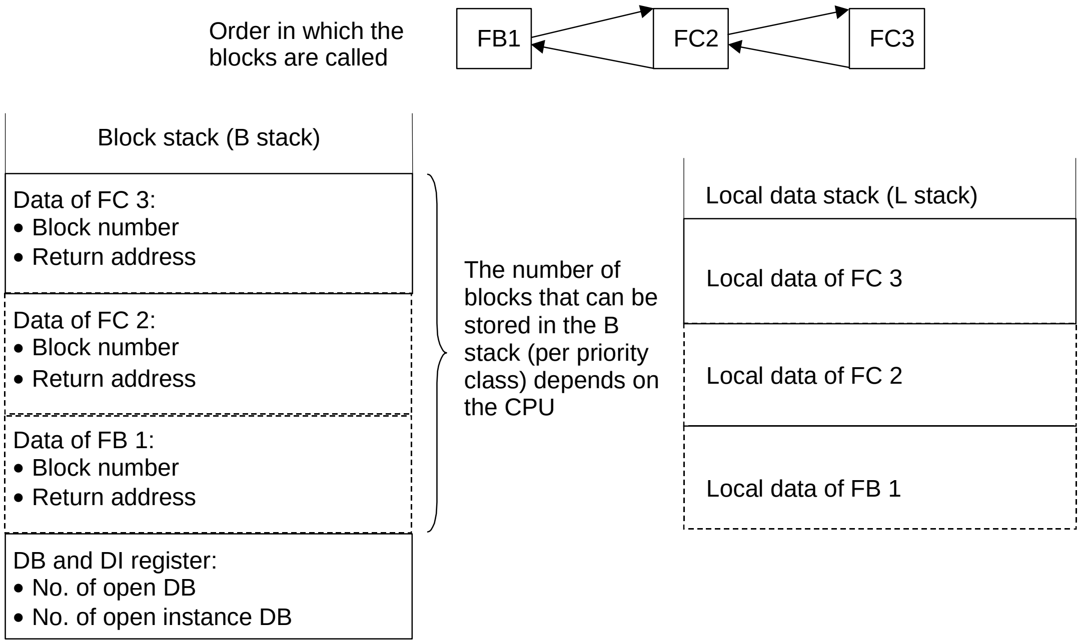
A.2.3.5 块堆栈

如果块的处理被另一个块的调用或更高优先级的(中断/错误检修)中断，B 堆栈存储下列数据：

- 编号、类型(OB、FB、FC、SFB、SFC)，并返回被中断块的地址。
- 当块被中断时，已打开的数据块(来自 DB 和 DI 寄存器)编号。

使用此数据，用户程序可以在中断后恢复。

如果 CPU 处于 STOP 模式，可以用 STEP 7 在编程设备上显示 B 堆栈。当 CPU 改变为 STOP 模式时，B 堆栈将列出所有没有完全执行的块。块以处理开始的顺序列出(参见下图)。



数据块寄存器

存在两个数据块寄存器。该寄存器组装有打开数据块的编号，如下：

- DB 寄存器包含打开的共享数据块的编号
- DI 寄存器包含打开的背景数据块的编号

A.2.3.6 诊断缓冲区

诊断缓存区按诊断信息出现的顺序进行显示。第一个条目包含最新的事件。诊断缓存区中的条目编号取决于模块及其当前工作模式。

诊断事件包括：

- 模块上的故障
- 过程接线中的错误
- CPU 中的系统错误
- CPU 上的模式转换
- 用户程序中的错误
- 用户自定义诊断事件(通过系统功能 SFC52)。

A.2.3.7 评估诊断缓冲区

系统状态表有一部分是诊断缓冲区，包含了以发生顺序排列的系统诊断事件和用户自定义的诊断事件的更多信息。当系统诊断事件发生时，输入诊断缓冲区的信息与传送到相应组织块的启动信息相同。

您无法清除诊断缓冲区中的这些条目，即使存储器复位，它们仍将保持。

诊断缓冲区为您提供了以下可能：

- 如果 CPU 切换到 STOP 模式，您可以评估最后导致 STOP 的事件，并定位原因。
- 快速检测错误原因，大大提高系统的可用性。
- 您可以评估和优化动态系统响应。

组织诊断缓冲区

诊断缓冲区设计为环形缓冲区的工作方式，这样可以利用条目的最大数目，它决定于单个模块。也就是说，当达到条目的最大数目时，下一个诊断缓冲区事件将导致最早的条目被删除。然后，所有的条目往后退一位。也就是说，最新的条目总是在诊断缓冲区的第一条。对于 S7300 CPU 314，可能的条目数目为 100：

诊断缓存区中显示的条目编号取决于模块及其当前工作模式。对于某些 CPU，可以设置诊断缓冲区的长度。

诊断缓冲区的内容

上部列表框中包含了所有诊断事件的列表，它们具有下列信息：

- 条目的序列号(最新的条目的编号为 1)
- 诊断事件的时间和日期：如果模块有集成的时钟，将显示模块的时间和日期。要使缓冲区中的时间数据有效，应设置模块的时间和日期并定期检查，这点很重要。
- 诊断事件的简短描述

在**下部**文本框中，将显示上部窗口的列表中所选事件的全部附加信息。这些信息包括：

- 事件编号
- 事件的描述
- 诊断事件引起的模式转换
- 在缓冲区中生成条目的，块中的错误的访问地址(块类型、块编号、相关地址)
- 已进入或已离开的事件状态
- 特定事件的附加信息

使用“事件帮助”按钮，可以在上部列表框中显示所选事件的附加信息。

关于事件标识号的信息可以在“系统块和系统功能的参考帮助”中找到(跳转到关于块、系统属性的语言描述与帮助)

以文本文件保存内容

使用“模块信息”对话框“诊断缓冲区”标签中的“另存为”按钮，可以以 ASCII 文本的格式保存诊断缓冲区的内容。

显示诊断缓冲区

通过“模块信息”对话框“诊断缓冲区”标签或者在程序中使用系统功能 SFC51 RDSYSST，可以在编程设备上显示诊断缓冲区的内容。

STOP 之前的最后一个条目

可以指定在从 RUN 切换到 STOP 前，将诊断缓冲区的最后一个条目自动发送到已登录的监视设备(例如，PG、OP、TD)，以更快速地定位和修复引起 STOP 的原因。

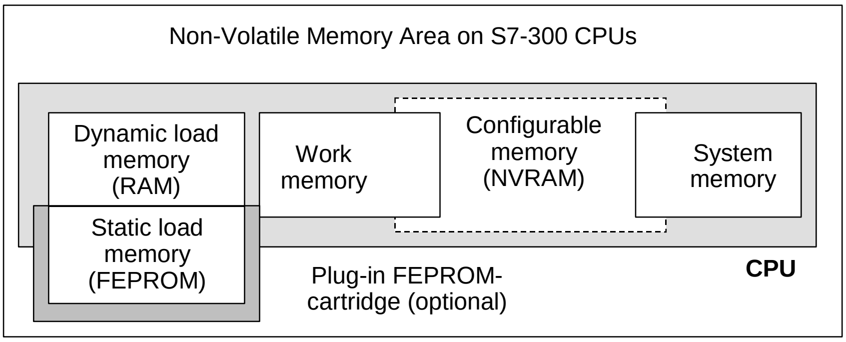
A.2.3.8 S7-300 CPU 上的保持存储器

如如果电源出现掉电或 CPU 存储器复位(MRES)的情况，S7-300 CPU 的存储器(动态装入存储器(RAM)、工作存储器和系统存储器)复位，而且这些区域先前包含的所有数据将丢失。对于 S7-300 CPU，可以用下列方法保护程序及其数据：

- 可以使用备用电池，保护在装入存储器、工作存储器中的所有数据和系统存储器的部分数据。
- 可以将程序存储在 EPROM 中(或者是存储卡或者是集成于 CPU 上，参见“S7-300 可编程控制器，硬件和安装”手册)。
- 依靠 CPU，在非易失性的 NVRAM 区中可以存储一定数量的数据。

使用 NVRAM

S7-300 CPU 在 NVRAM (非易失性 RAM)中提供一个区域(参见下图)。如果已在装入存储器的 EPROM 中存储程序，从而可以通过组态 CPU 保存某些数据(在有电源掉电或当 CPU 从 STOP 切换到 RUN 的情况下)。



为此，设置 CPU 以便下列数据保存在非易失性 RAM 中：

- 数据包含在 DB 中(只有将程序已存储在装入存储器的 EPROM 中时才有效)
- 定时器和计数器值
- 保存在位存储器的数据。

在每个 CPU 上，可以保存一定数量的定时器、计数器和存储位。一定数量的字节也可用，包含在 DB 内的数据可以保存在那里。

CPU 的 MPI 地址存储在 NVRAM 中。确保在电源掉电或存储器复位后，CPU 能够进行通讯。

使用备用电池保护数据

通过使用备用电池，在电源掉电期间，装入存储器和工作存储器将保持。如果组态 CPU，使得定时器、计数器和位存储器保存在 NVRAM 中，不管是否使用备用电池，此信息也将保留。

组态 NVRAM 的数据

当使用 STEP 7 组态 CPU 时，可以决定哪些存储器区将保持。

可以在 NVRAM 中组态的存储器数量取决于正在使用的 CPU。不能备份比 CPU 指定的数目多的数据。

A.2.3.9 S7-400 CPU 上保持存储器

没有备用电池的操作

如果使用没有备用电池时操作系统，当发生电源掉电或复位 CPU 存储器(MRES)时，S7-400 CPU 的存储器(动态装入存储器(RAM)、工作存储器和系统存储器)复位，并且丢失所有包含在这些区域的数据。

没有备用电池，只能重新启动(暖重启)，且没有保持的存储器区。随着电源掉电，只有 MPI 参数(例如，CPU 的 MPI 地址)保留。这意味着在电源掉电或存储器复位后，CPU 仍能够进行通讯。

有备用电池的操作

如果使用电池备份存储器：

- 在电源掉电后，当 CPU 重新启动时，所有 RAM 区域的全部内容都保留。
- 在重启动期间(暖重启)，位存储器、定时器和计数器的地址区域被清零。保留数据块的内容。
- 除了那些被设计为非保持的位存储器、定时器和计数器之外，RAM 工作存储器的内容也保留。

组态保持数据区

可以声明一定数目的存储位、定时器和计数器为保持(数目取决于 CPU)。在重启动期间(暖重启)，当使用备用电池时，此数据也保留。

当用 STEP 7 分配参数时，可定义在重启动(暖重启)期间哪些存储位、定时器和计数器应该保持。只能备份 CPU 允许数量的数据。

关于更详细的有关定义保持存储器区的信息，请参见“S7-400、M7-400 可编程控制器，模块技术规范”参考手册。

工作存储器中的可组态存储器对象

对于某些 CPU，对象的大小，如本地数据或诊断缓存区，可以在“组态硬件”下设置。例如，如果减少预设值的数量，将留出工作存储器的更多部分作为它用。在“模块信息”对话框的“存储器”标签中找到这些 CPU 的设置(“详细资料”按钮)。

当改变存储器组态和下载新的组态到可编程控制器时，需要冷重启 CPU，以使改变生效。

A.2.3.10 工作存储器中的可组态存储器对象

对于某些 CPU，对象的大小，如本地或诊断缓存区，可以在硬件组态中设置。例如，如果减少缺省值，将留出工作存储器的更多部分作为它用。将在模块信息的“存储器”标签中显示这些 CPU 的设置(“详细资料”按钮)。

在存储器组态改变和下载到可编程控制器后，必须执行冷启动，以使改变生效。

A.3 数据类型和参数类型

A.3.1 数据类型和参数类型的引言

用户程序中的所有数据必须被数据类型识别。下列数据类型可用：

- STEP 7 提供的基本数据类型
- 用户可以通过组合基本数据类型创建复杂数据类型
- 用来定义传送到 FB 或 FC 参数的参数类型

常规信息

语句表、梯形图和功能块图指令使用特定长度的数据对象。例如，位逻辑指令使用位。装载和传递指令(STL)以及移动指令(LAD 和 FBD)使用字节、字和双字。

位是二进制的数字“0”或“1”。一个字节由 8 位组成，一个字由 16 位组成，双字由 32 位组成。

数学运算指令也使用字节、字或双字。在这些字节、字或双字地址中，可以对各种格式，如整数和浮点数，进行编码。

当使用符号寻址时，定义符号并指定这些符号的数据类型(参见下表)。不同的数据类型具有不同格式选项和计数法。

此章只描述一些书写编号和常数的方法。下表列出的编号和常数的格式，不再详细解释。

格式	以位计的长度	计数法
十六进制	8、16 和 32	B#16#、W#16#和 DW#16#
二进制	8、16 和 32	2#
IEC 日期	16	D#
IEC 时间	32	T#
时间	32	TOD#
字符	8	‘A’

A.3.2 基本数据类型

A.3.2.1 基本数据类型

每个基本数据类型具有定义的长度。下表列出了基本数据类型。

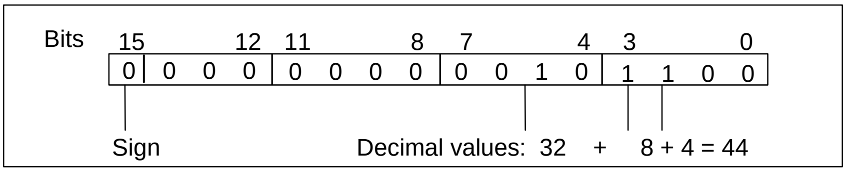
类型和描述	以位计的长度	格式选项	范围和计数法(最低到最高值)	实例
BOOL(位)	1	布尔文本	TRUE/FALSE	TRUE
BYTE (字节)	8	十六进制的数字	B#16#0 到 B#16#FF	L B#16#10 L byte#16#10
WORD (字)	16	二进制的数字 十六进制的数字 BCD 十进制无符号数字	2#0 到 2#1111_1111_1111_1111 W#16#0 到 W#16#FFFF C#0 到 C#999 B#(0.0)到 B#(255.255)	L 2#0001_0000_0000_0000 L W#16#1000 L word#16#1000 L C#998 L B#(10,20) L byte#(10,20)
DWORD (双字)	32	二进制的数字 十六进制的数字 十进制无符号数字	2#0 到 2#1111_1111_1111_1111 1111_1111_1111_1111 DW#16#0000_0000 到 DW#16#FFFF_FFFF B#(0,0,0,0)到 B#(255,255,255,255)	2#1000_0001_0001_1000_ 1011_1011_0111_1111 L DW#16#00A2_1234 L dword#16#00A2_1234 L B#(1, 14, 100, 120) L byte#(1,14,100,120)
INT (整数)	16	十进制有符号数字	-32768 - 32767	L 1
DINT (整数，32 位)	32	十进制有符号数字	L#-2147483648 到 L#2147483647	L L#1
REAL (浮点数)	32	IEEE 浮点数	上限：±3.402823e+38 下限：±1.175 495e-38	L 1.234567e+13
S5TIME (SIMATIC 时间)	16	S7 时间 以步长 10 毫秒(缺省值)	S5T#0H_0M_0S_10MS 到 S5T#2H_46M_30S_0MS 和 S5T#0H_0M_0S_0MS	L S5T#0H_1M_0S_0MS L S5TIME#0H_1H_1M_0S_0MS
TIME (IEC 时间)	32	IEC 时间步长为 1 毫秒，有符号整数	- T#24D_20H_31M_23S_648MS 到 T#24D_20H_31M_23S_647MS	L T#0D_1H_1M_0S_0MS L TIME#0D_1H_1M_0S_0MS
DATE (IEC 日期)	16	IEC 日期步长为 1 天	D#1990-1-1 到 D#2168-12-31	L D#1996-3-15 L DATE#1996-3-15
TIME_OF_DAY (时间)	32	时间步长为 1 毫秒	TOD#0:0:0.0 到 TOD#23:59:59.999	L TOD#1:10:3.3 L TIME_OF_DAY#1:10:3.3
CHAR (字符)	8	ASCII 字符	'A'、'B'等等	L ‘E’

A.3.2.2 数据类型 INT 的格式(16 位整数)

整数有一个符号，指出它是正整数还是负整数。一个整数(16 位)在存储器中占用的空间是一个字。下表给出整数(16 位)的范围。

格式	范围
整数(16 位)	-32 768 至+32 767

下图中将整数+44 显示为二进制的数字。

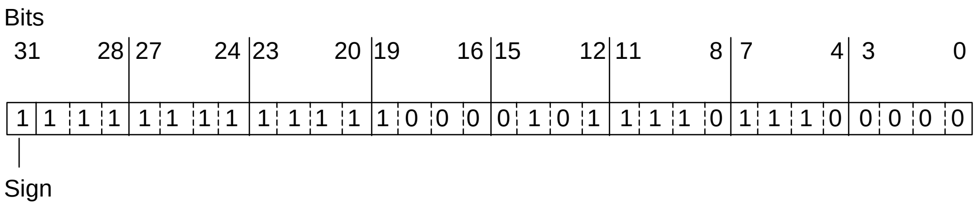


A.3.2.3 数据类型 DINT 的格式(32 位整数)

整数有一个符号，指出它是正整数还是负整数。双整数在存储器中占用的空间是两个字。下表给出双整数的范围。

格式	范围
整数(32 位)	-2 147 483 648 到+2 147 483 647

下图中将整数-500 000 显示为二进制的数字。在二进制系统中，整数的负数形式表示为正整数的二进制补码。通过取反所有位的状态，然后将结果加+1，以获取整数的二进制补码。



A.3.2.4 数据类型 REAL 的格式(浮点数)

浮点格式数字表示的通用形式是“数字 = m * b 的 E 次方”。基数“b”和指数“E”是整数；尾数“m”是有理数。

这种类型的数字表达法的优点在于：在有限的空间内能够表示非常大和非常小的数值。在尾数和指数的有限位数内，可以覆盖很大范围的数字。

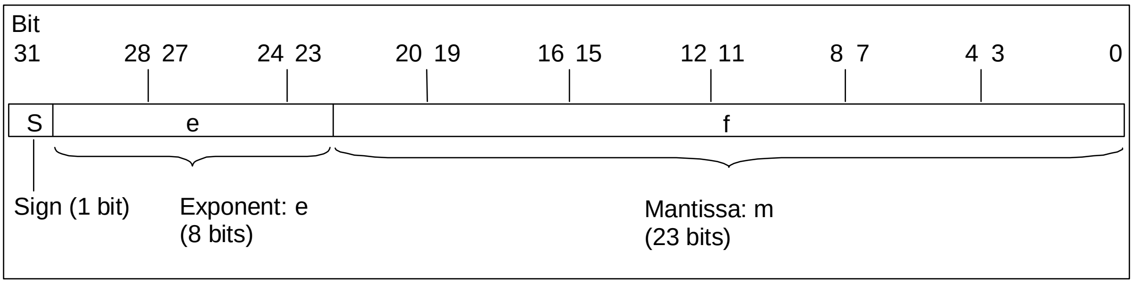
缺点是计算的精度有限。例如，当构成两个数字的和时，指数必须通过移位尾数(移动小数点)来匹配，因为只有具有相同指数的数字才能相加。

STEP 7 中的浮点数格式

STEP 7 中的浮点数符合基本格式，单宽度，在 ANSI/IEEE 标准 754-1985，二进制浮点算术的 IEEE 标准中描述的。它们由下列组件组成：

- 符号 S
- 指数 e = E + 偏差，以常数增加(偏差 = +127)
- 尾数 m 的小数部分。
  尾数的整个数字部分不和其它数字一起存储，因为在有效数字范围内，它总是等于 1。

这三个组成部分一起占用一个双字(32 位)：



下表显示浮点格式中各个位的值。

浮点数的组成部分	位号	值
符号 S	31	
指数 e	30	2 的 7 次方
...	...	...
指数 e	24	2 的 1 次方
指数 e	23	2 的 0 次方
尾数 m	22	2 的-1 次方
...	...	...
尾数 m	1	2 的-22 次方
尾数 m	0	2 的-23 次方

使用三个组成部分 S、e 和 m，以此形式表示的数值由以下公式定义：

数字 = 1.m ∗ 2 的(e - 偏差)次方

此处：

- e：1 ≤ e ≤ 254
- 偏差：偏差 = 127。这意味着不需为指数附加符号。
- S：对于正数，S = 0；对于负数，S = 1。

浮点数的数值范围

使用上述的浮点格式，下列结果为：

- 最小的浮点数 = 1.0 ∗ 2 的(1-127)次方 = 1.0 ∗ 2 的(-126)次方
  = 1.175 495E-38，
- 最大的浮点数 = 2-2 的(-23)次方 ∗ 2 的(254-127)次方 = 2-2 的(-23)次方 ∗ 2 的(+127)次方
  = 3.402 823E+38

数字零表示为 e = m = 0；e = 255 和 m = 0 表示“无限大”。

格式	范围
浮点数依据 ANSI/IEEE 标准	-3.402 823E+38 至-1.175495E-38 和 0 和+1.175 495E-38 至+3.402 823E+38

下表给出了对于不在有效范围内的浮点数的指令结果状态字中符号位的状态。

结果的无效范围	CC1	CC0	OV	OS
-1.175494E-38 < 指令结果 < -1.401298E-45(负数)下溢	0	0	1	1
+1.401298E-45 < 指令结果 < +1.175494E-38(正数)下溢	0	0	1	1
指令结果 < -3.402823E+38 (负数)上溢	0	1	1	1
指令结果 > 3.402823E+38 (正数)上溢	1	0	1	1
不是有效的浮点数或无效指令(输入值超出有效的值范围)	1	1	1	1

使用数学运算时的注意事项：

例如，当尝试求-2 的平方根时，将获得结果“不是有效的浮点数”。因此在基于结果继续计算之前，始终应该先估计数学运算中的状态位。

修改变量时的注意事项：

例如，如果用于浮点运算的值存储在存储器双字中，可以用任何位模式修改这些值。然而，不是每个位模式都是有效的数字。

计算浮点数时的精度

警告

涉及包含非常大和非常小数字的一长串数值的计算，可能会导致不精确的结果。

STEP 7 中的浮点数精确到 6 位小数。因此当输入浮点常数时，最多只能指定 6 位小数。

注意

计算的精度为 6 个小数位数意味着，例如加法 number1 + number2 = number1 如果 number1 大于 number2 ∗ 10 的 y 次方，y>6：

100 000 000 + 1 = 100 000 000。

浮点格式数字的实例

下图给出了下列十进制值的浮点格式：

- 10.0
- Pi (3.141593)
- 2 的平方根(1.414214)

在第一个实例中的数字 10.0 从其浮点格式得出(十六进制的表达式：4120 0000)如下：

e = 2 的 7 次方 + 2 的 1 次方 = 2 + 128 = 130

m = 2 的(-2)次方 = 0.25

这导致：

(1 + m) ∗ 2 的(e - 偏差)次方 = 1.25 ∗ 2 的 3 次方 = 10.0

[1.25 ∗ 2 的(130-127)次方 = 1.25 ∗ 2 的 3 次方 = 10.0]

Decimal value 10.0

Hexadecimal value	4	1	2	0	0	0	0	0
Bits	31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
	0 1 0 0	0 0 0 1	0 0 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

Sign of Mantissa: s (1 bit)

Exponent: e (8 bits)

Mantissa: m (23 bits)

e = 27 + 21 = 130

f = 2-2 = 0.25

1.f _ 2e-bias = 1.25 _ 23 = 10.0

[1.25 _ 2(130-127) = 1.25 _ 23 = 10.0]

Decimal value 3.141593

Hexadecimal value	4	0	4	9	0	F	D	C
Bits	31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
	0 1 0 0	0 0 0 0	0 1 0 0	1 0 0 1	0 0 0 0	1 1 1 1	1 1 0 1	1 1 0 0

Sign of Mantissa: s (1 bit)

Exponent: e (8 bits)

Mantissa: m (23 bits)

Decimal value 1.414214

Hexadecimal value	3	F	B	5	0	4	F	7
Bits	31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
	0 0 1 1	1 1 1 1	1 0 1 1	0 1 0 1	0 0 0 0	0 1 0 0	1 1 1 1	0 1 1 1

Sign of Mantissa: s (1 bit)

Exponent: e (8 bits)

Mantissa: m (23 bits)

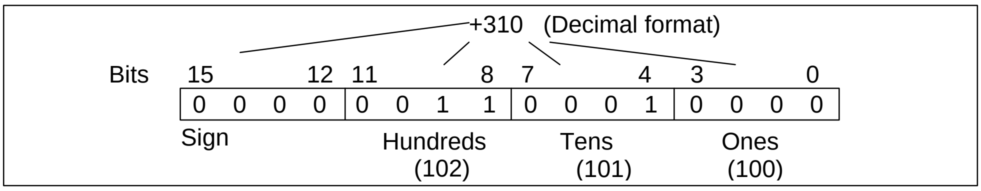
A.3.2.5 以二进制编码的十进制数字中的数据类型 WORD 和 DWORD 的格式

二进制编码的十进制的(BCD)格式通过用一组二进制的数字(位)来表示十进制数字。一组 4 个位表示一个有符号的十进制数字的一位或十进制数字的符号。4 位一组的组合形成字(16 位)或双字(32 位)。四个最高有效位指示数字的符号(1111 表示负的，0000 表示正的)。具有 BCD 编码地址的命令只求最高位的值(字格式中是第 15 位，双字格式中是第 31 位)。下表给出了两种类型 BCD 数字的格式和范围。

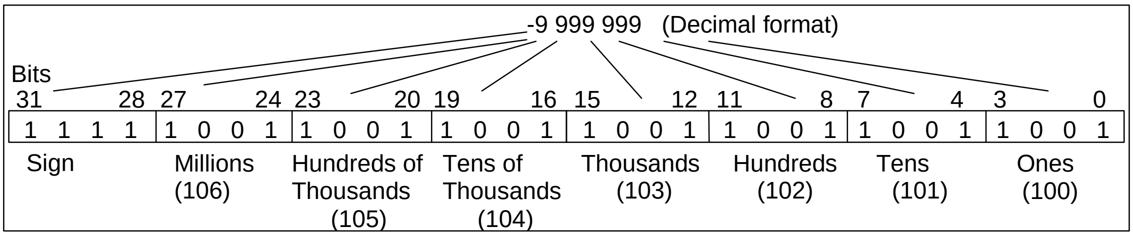
格式	范围
字 (位，带符号的三位 BCD 数字)	-999 至+999
双字 (32 位，带符号的七位 BCD 数字)	-9 999 999 至+9 999 999

下图给出了以下格式的二进制编码的十进制数字的实例：

- 字格式

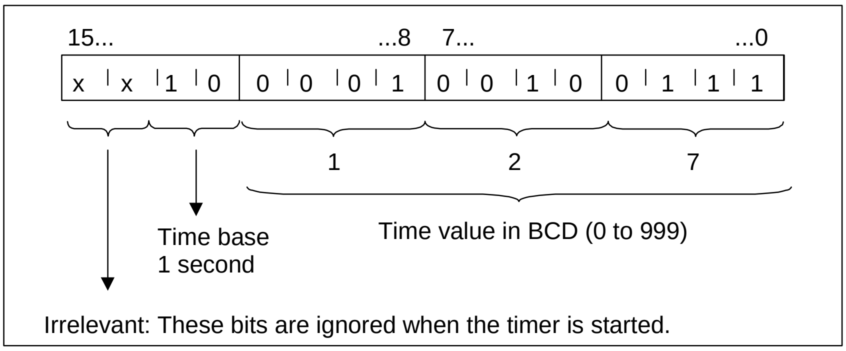


- 双字格式



A.3.2.6 数据类型 S5TIME 的格式(持续时间)

当使用 S5TIME 数据类型输入持续时间时，输入条目以二进制编码的十进制格式进行存储。下图说明了时间值为 127，时间基准为 1s 的时间地址的内容。



当使用 S5TIME 时，输入时间值的范围为 0 - 999，并说明时间基准(参见下表)。时间基准指的是定时器以多少间隔为一个单位来递减时间值，一直到达 0 的时间间隔。

S5TIME 的时间基准

时间基准	时间基准的二进制代码
10ms	00
100ms	01
1s	10
10s	11

可以使用下列语法格式之一预装载时间值：

- L[1] W#16#wxyz
  - 此处 w = 时间基准(即时间间隔或分辨率)
  - 其中 xyz = 以二进制编码的十进制格式表示的时间值
- L[1] S5T#aH_bbM_ccS_dddMS
  - 此处 a = 小时，bb = 分钟，cc = 秒，dd = 毫秒
  - 时间基准自动选择，该值四舍五入至具有此时间基准的下一个较小的数字。

可以输入的最大时间值是 9,990 秒或 2 小时_46 分钟_30 秒。

[1] = L 只能在 STL 编程中指定

A.3.3 复杂数据类型

A.3.3.1 复杂数据类型

复杂数据类型定义大于 32 位的数字数据群或包含其它数据类型的数据群。STEP 7 允许下列复杂数据类型：

- DATE_AND_TIME
- STRING
- ARRAY
- STRUCT
- UDT (用户自定义数据类型)
- FB 和 SFB

下表中描述了复杂数据类型。要么在逻辑块的变量说明中，要么在数据块中定义结构和数组。

数据类型	描述
DATE_AND_TIME DT	定义具有 64 位(8 个字节)的区域。此数据类型以二进制编码的十进制的格式保存：
STRING	定义最多有 254 个字符的组(数据类型 CHAR)。为字符串保留的标准区域是 256 个字节长。这是保存 254 个字符和 2 个字节的标题所需要的空间。可以通过定义即将存储在字符串中的字符数目来减少字符串所需要的存储空间(例如：string[9]'Siemens')。
ARRAY	定义一个数据类型(基本或复杂)的多维组群。例如：“ARRAY [1..2,1..3] OF INT”定义 2 x 3 的整数数组。使用下标(“[2,2]”)访问数组中存储的数据。最多可以定义 6 维数组。下标可以是任何整数(-32768 - 32767)。
STRUCT	定义一个数据类型任意组合的组群。例如，可以定义结构的数组或结构和数组的结构。
UDT	在创建数据块或在变量声明中声明变量时，简化大量数据的结构化和数据类型的输入。在 STEP 7 中，可以组合复杂的和基本的数据类型以创建用户的“用户自定义”数据类型。UDT 具有自己的名称，因此可以多次使用。
FB、SFB	确定分配的背景数据块的结构，并允许在一个背景数据块中传送数个 FB 调用的背景数据。

保存结构化的数据类型和字的限制是一致的(WORD 对齐)。

A.3.3.2 数据类型 DATE_AND_TIME 的格式

当使用 DATE_AND_TIME 数据类型(DT)输入日期和时间时，输入条目以 8 个字节的二进制编码的十进制格式存储。DATE_AND_TIME 数据类型的范围如下：

DT#1990-1-1-0:0:0.0 到 DT#2089-12-31-23:59:59.999

下面的实例给出了日期和时间分别是 1993 年 12 月 25 日星期四上午 8:01 和 1.23 秒的语法。下列两种格式是可行的：

- DATE_AND_TIME#1993-12-25-8:01:1.23
- DT#1993-12-25-8:01:1.23

如下的专用 IEC(国际电工委员会)标准功能在使用 DATE_AND_TIME 数据类型时可用：

- 转变日期和日时钟到 DATE_AND_TIME 格式

  FC3: D_TOD_DT
- 从 DATE_AND_TIME 格式提取日期

  FC6: DT_DATE
- 从 DATE_AND_TIME 格式提取星期

  FC7: DT_DAY
- 从 DATE_AND_TIME 格式提取时间

  FC8: DT_TOD

下表给出了包含日期和时间信息的字节的内容，例子为 1993 年 12 月 25 日星期四上午 8:01 和 1.23 秒。

字节	目录	实例
0	年	B#16#93
1	月	B#16#12
2	日	B#16#25
3	小时	B#16#08
4	分钟	B#16#01
5	秒	B#16#01
6	MSEC 的两个最高有效位	B#16#23
7 (4MSB)	MSEC 的两个最低有效位	B#16#0
7 (4LSB)	星期 1 = 星期日 2 = 星期一 ... 7 = 星期六	B#16#5

数据类型 DATE_AND_TIME 的允许范围是：

- 最小：DT#1990-1-1-0:0:0.0
- 最大：DT#2089-12-31-23:59:59.999

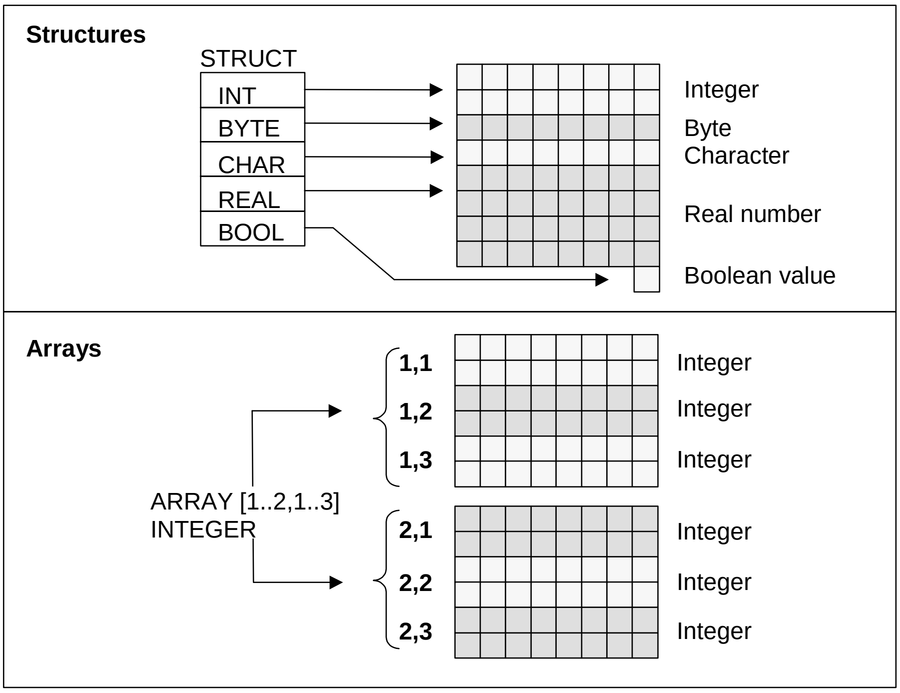
	可能的值范围	BCD 代码
年	1990 - 1999 2000 - 2089	90h - 99h 00h - 89h
月	1 - 12	01h - 12h
日	1 - 31	01h - 31h
小时	00 - 23	00h - 23h
分钟	00 - 59	00h - 59h
秒	00 - 59	00h - 59h
毫秒	0 - 999	000h - 999h
星期	星期日 - 星期六	1h - 7h

A.3.3.3 使用复杂数据类型

可以通过组合基本的和复杂的数据类型创建如下的复杂数据类型，用于创建新的数据类型：

- 数组(数据类型 ARRAY)：数组与相同类型的数据群组合形成单个单元。
- 结构(数据类型 STRUCT)：结构与不同的数据类型组合形成单个单元。
- 字符串(数据类型 STRING)：字符串定义具有最多 254 个字符的一维数组(数据类型 CHAR)。字符串只能作为一个单元传送。字符串的长度必须匹配块的形式参数和实际参数。
- 日期和时间(数据类型 DATE_AND_TIME)：日期和时间数据类型存储年、月、日、小时、分钟、秒、毫秒和星期。

下图给出了数组和结构如何在一个区域内结构化数据类型和保存信息。在 DB 中，或者在 FB、OB 或 FC 的变量声明中定义数组或结构。



A.3.3.4 使用数组访问数据

数组

数组组合一组相同的数据类型(基本或复杂)以构成单元。可以创建包含数组的数组。当定义数组时，必须做如下步骤：

- 给数组指定名称。
- 用关键字 ARRAY 声明数组。
- 使用下标指定数组的大小。指定数组中各个维(最多为 6 维)的第一个和最后一个数字。将下标输入方括号中，每个维数之间用逗号隔开，维数中的第一个数字和最一个数字之间用两个点隔开。例如，下列下标定义一个三维数组：

  [1..5,-2..3,30..32]

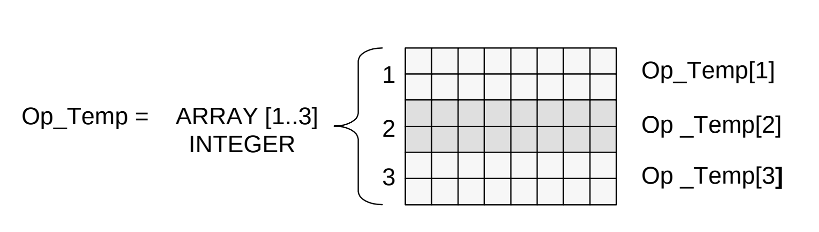
- 指定包含在数组中的数据的类型。

实例：1

下图显示具有三个整数的数组。使用下标访问数组中存储的数据。下标是在方括号中的数字。例如，第二个整数的下标是 Op_temp[2]。

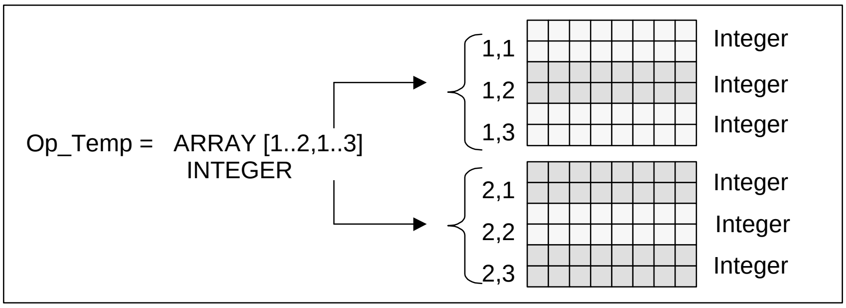
下标可以是任何整数(-32768 - 32767)，包括负的值。下图中的数组也可以被定义为 ARRAY [-1..1]。然后，第一个整数的下标可以是 Op_temp[-1]，第二个可以是 Op_temp[0]，而第三个整数可以是 Op_temp[1]。

Address	Name	Type	Init. Value	Comment
0.0		STRUCT		
+0.0	Op_Temp	ARRAY[1..3]		
*2.0		INT		
=3.0		END_STRUCT		



实例 2

数组也可以描述多维数据类型的组。下图给出两维整数数组。



使用下标访问多维数组中的数据。在此例中，第一个整数是 Op_temp[1,1]，第三个是 Op_temp[1,3]，第四个是 Op_temp[2,1]，第六个是 Op_temp[2,3]。

可以为数组定义最多 6 维(6 个下标)。例如，可以定义变量 Op_temp 如下作为六维数组：

ARRAY [1..3,1..2,1..3,1..4,1..3,1..4]

在此数组中的第一个元素的下标是 Op_temp[1,1,1,1,1,1]。最后元素的下标是 Op_temp[3,2,3,4,3,4]。

创建数组

当在 DB 中或在变量声明中声明数据时，可以定义数组。当声明数组时，在方括号中数组大小后跟指定的关键字(ARRAY)，如下：

[下限值..上限值]

在多维数组中，也可以指定另外的上限和下限值，并用逗号分开各个维数。下图显示格式为 2 x 3 的数组创建的声明。

Address	Name	Type	Init. Value	Comment
0.0		STRUCT		
+0.0	Heat_2x3	ARRAY[1..2,1..3]		
*2.0		INT		
=6.0		END_STRUCT		

输入数组的初始值

当创建数组时，可以将初始值赋给每个数组元素。STEP 7 提供两种方式输入初始值：

- 输入单个值：对于数组的每个元素，指定有效的数组数据类型值。.按元素顺序指定值：[1,1]。记住各个元素必须用逗号互相隔开。
- 指定重复因子：对于具有相同初始值的有序元素，可以指定元素的数目(重复因子)，以及这些元素的初始值。输入重复因子的格式是 *x(y)*，此处 *x* 是重复因子，而 *y* 是重复的值。

如果使用上图中声明的数组，可以如下为全部六个元素指定初始值：17、23、-45、556、3342、0。也可以通过指定 6(10)设置所有六个元素的初始值为 10。可以指定前两个元素的特定值，然后设置余下的四个元素为 0，如下指定： 17、23、4(0)。

访问数组中的数据

通过数组中特定元素的下标访问数组中的数据。下标使用符号名。

实例：如果在上图中声明的数组以 DB20(电机)的第一个字节开始，用下列地址访问数组的第二个元素：

Motor.Heat_2x3[1,2].

将数组作为参数

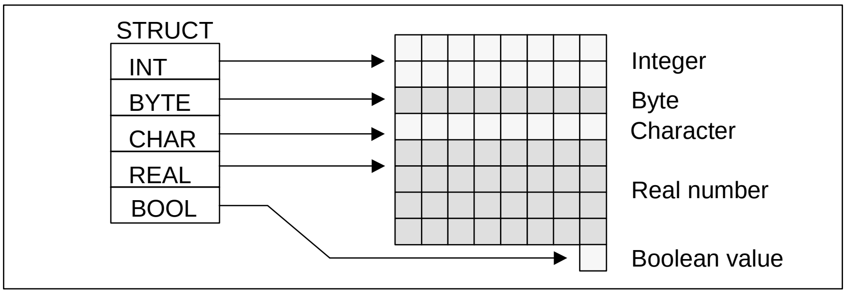
可以将数组作为参数传送。如果在变量声明中参数被声明为 ARRAY，必须传送整个数组(而不是单个元素)。然而，当调用块时，倘若数组的元素符合参数的数据类型，数组的元素就可以分配给一个参数。

如果使用数组作为参数，数组不需要具有相同的名称(它们甚至不需要名称)。然而，两个数组(形式参数和实际参数)必须具有相同的结构。例如，只有当块的形参定义为格式 2 x 3 的整数数组，并且调用操作提供的实际参数为 2 x 3 的整数数组时，格式 2 x 3 的整数数组才能作为参数传送。

A.3.3.5 使用结构访问数据

结构

结构组合了各种数据类型(基本和复杂数据类型，包括数组和结构)，形成一个单元。可以归类数据以适合过程控制。因此，也可以将参数作为数据单元传送，而不是作为单个元素传送。下图说明包含整数、字节、字符、浮点数和布尔值的结构。



结构最多可以嵌套 8 层(例如，结构由包含数组的结构组成)。

创建结构

当在 DB 中或在逻辑块的变量声明中声明数据时，就定义了结构。

下图说明了结构的声明(*Stack_1*)包含下列元素：整数(用作保存总数)、字节(用于保存原始数据)、字符(用于保存控制代码)、浮点数(用于保存温度)和布尔型存储位(用于终止信号)。

Address	Name	Type	Init. Value	Comment
0.0	Stack_1	STRUCT		
+0.0	Amount	INT	100	
+2.0	Original_data	BYTE		
+4.0	Control_code	CHAR		
+6.0	Temperature	REAL	120	
+8.1	End	BOOL	FALSE	
=10.0		END_STRUCT		

为结构赋初始值

如果想要将初始值赋给结构的每个元素，则要指定有效数据类型的值和元素名称。例如可以赋值下列初始值(给上图中声明的结构)：

Amount	=	100
Original_data	=	B#(0)
Control_code	=	'C'
Temperature	=	120
End	=	False

在结构中保存和访问数据

用户能够访问结构的各个元素。可以使用符号地址(例如，*Stack_1.Temperature*)。然而，也可以指定元素所位于的绝对地址(实例：如果 Stack_1 位于 DB20，以字节 0 开始，*amount* 的绝对地址是 *DB20.DBW0*，而 *temperature* 的地址是 *DB20.DBD6*)。

使用结构作为参数

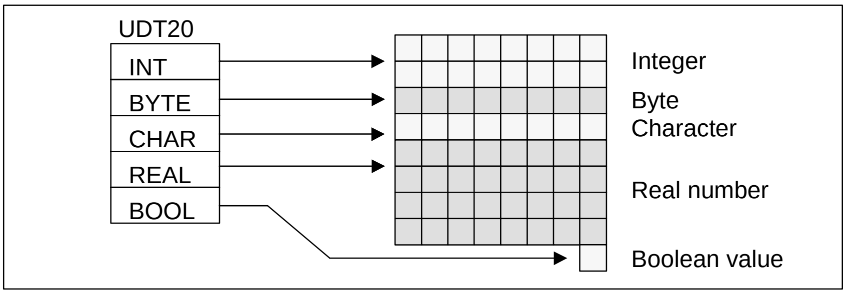
可以将结构作为参数传送。如果参数在变量声明中被声明为 STRUCT，则必须传送具有相同组件的结构。然而，当调用块时，倘若结构的元素符合参数的数据类型，结构的元素就可以分配给一个参数。

如果使用结构作为参数，两个结构(对于形式参数和实际参数的)必须具有相同的组件。换句话说，相同的数据类型必须以相同的顺序排列。

A.3.3.6 使用用户自定义数据类型访问数据

用户自定义数据类型

用户自定义数据类型(UDT)可以组合基本和复杂数据类型。可以指定名称给 UDT，并多次使用它们。下图说明包含整数、字节、字符、浮点数和布尔值的用户自定义数据类型的结构。



代替逐一地输入所有数据类型或作为结构，只需要指定“UDT20”作为数据类型，STEP 7 自动地分配相应的内存空间。

创建用户自定义数据类型

使用 STEP 7 定义 UDT。下图中给出的 UDT 包括如下的元素：整数(用作保存总数)、字节(用于保存原始数据)、字符(用于保存控制代码)、浮点数(用于保存温度)和布尔型存储位(用于终止信号)。可以在符号表中指定一个符号名称给 UDT (例如 *process data*)。

Address	Name	Type	Init. Value	Comment
0.0	Stack_1	STRUCT		
+0.0	Amount	INT	100	
+2.0	Original_data	BYTE		
+4.0	Control_code	CHAR		
+6.0	Temperature	REAL	120	
+8.1	End	BOOL	FALSE	
=10.0		END_STRUCT		

一旦已创建 UDT，可以象数据类型那样使用 UDT。例如，如果在 DB 中(或在 FB 的变量说明中)为变量声明了数据类型 UDT200。

下图给出了具有变量 *process_data_1* 以及数据类型 *UDT200 的 DB*。只指定 *UDT200* 和 *process_data_1*。当编译 DB 时，以斜体字显示被创建的数组。

Address	Name	Type	Init. Value	Comment
0.0		STRUCT		
+6.0	Process_data_1	UDT200		
=6.0		END_STRUCT		

为用户自定义数据类型分配初始值

如果希望将初始值赋给用户自定义数据类型的每个元素，要指定有效数据类型的值和元素的名称。例如可以赋值下列初始值(给上图中声明的用户自定义数据类型)：

Amount	=	100
Original_data	=	B#16#0)
Control_code	=	'C'
Temperature	=	1.200000e+002
End	=	False

如果声明变量为 UDT，变量的初始值是当创建 UDT 时指定的值。

在用户自定义数据类型中保存和访问数据

用户访问 UDT 的各个元素。可以使用符号地址(例如，*Stack_1.Temperature*)。然而，也可以指定元素位于的绝对地址(实例：如果 *Stack_1* 位于 DB20，以字节 0 开始，amount 的绝对地址是 *DB20.DBW0*，而 temperature 的地址是 *DB20.DBD6*)。

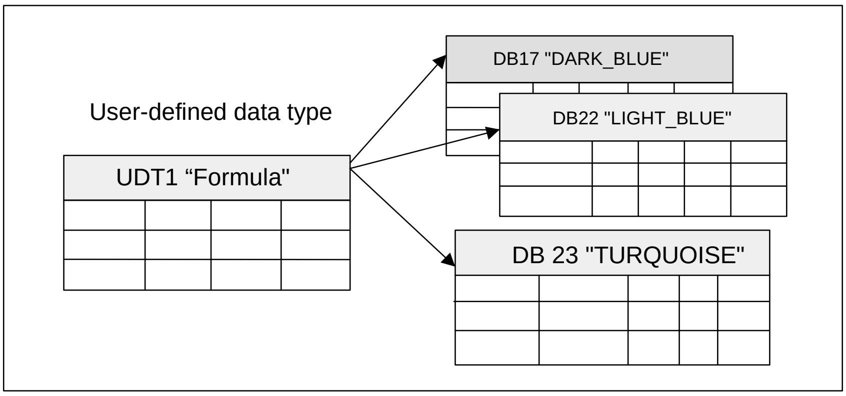
使用用户自定义数据类型作为参数

可以将数据类型 UDT 的变量作为参数传送。如果参数在变量说明中被声明为 UDT，必须传送具有相同结构的 UDT。然而，当调用块时，倘若 UDT 的元素符合该参数的数据类型，UDT 的元素可以分配给一个参数。

具有分配的 **UDT** 的 **DB** 优点

通过使用一次创建的 UDT，可以生成具有相同数据结构的大量数据块。然后就可以使用这些数据块为特定的任务输入不同的实际值。

例如，如果为公式构造一个 UDT (如为混和颜色)，可以将此 UDT 分配给每个包含不同数量的数个 DB。



数据块的结构由分配给它的 UDT 确定。

A.3.4 参数类型

A.3.4.1 参数类型

除了基本和复杂数据类型外，也可以为块之间传送的形式参数定义参数类型。STEP 7 识别下列参数类型：

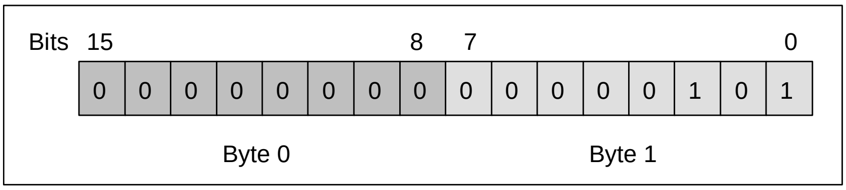
- TIMER 或 COUNTER：指定当执行块时将使用的特定定时器或特定计数器。如果赋值给 TIMER 或 COUNTER 参数类型的形参，相应的实际参数必须是定时器或计数器，换句话说，在正整数之后输入“T”或“C”。
- 块：指定用作输入或输出的特定块。参数的声明确定使用的块类型(FB、FC、DB 等)。如果提赋给 BLOCK 参数类型的形参，指定块地址作为实际参数。实例: “FC101”(当使用绝对寻址时)或“Valve”(使用符号寻址)。
- POINTER：参考变量的地址。指针包含地址而不是值。当赋值给 POINTER 参数类型的形式参数，指定地址作为实际参数。在 STEP 7 中，可以用指针格式或简单地以地址指定指针(例如，M 50.0)。寻址以 M 50.0 开始的数据的指针格式的实例：P#M50.0
- ANY：当实际参数的数据类型未知或当可以使用任何数据类型时，可以使用这个。关于 ANY 参数类型的更多信息，参见章节“参数类型 ANY 的格式”和“使用参数类型 ANY”。

参数类型也可以在用户自定义数据类型(UDT)中使用。关于 UDT 的更多信息，参见章节“使用用户自定义数据类型以访问数据”。

参数	容量	描述
TIMER	2 个字节	指示程序在调用的逻辑块中使用的定时器。 格式： T1
COUNTER	2 个字节	指示程序在调用的逻辑块中使用的计数器。 格式： C10
BLOCK_FB BLOCK_FC BLOCK_DB BLOCK_SDB	2 个字节	指示程序在调用的逻辑块中使用的块。 格式： FC101 DB42
POINTER	6 个字节	识别地址。 格式：P#M50.0
ANY	10 个字节	在当前参数的数据类型未知时使用。 格式： P#M50.0 BYTE 10 数据类型的 ANY 格式 P#M100.0 WORD 5 L#1COUNTER 10 数据类型的 ANY 格式

A.3.4.2 参数类型 BLOCK、COUNTER、TIMER 的格式

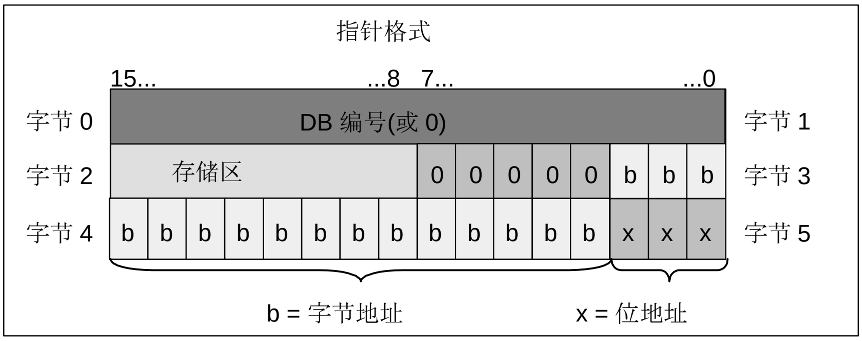
STEP 7 将参数类型 BLOCK、COUNTER 和 TIMER 以二进制数字存储在字(32 位)中。下图给出这些参数类型的格式。



块、定时器和计数器允许的数目取决于 S7 CPU 的类型。在“S7-300 可编程控制器、硬件和安装手册”或“S7-400、M7-400 可编程控制器、硬件和安装手册”中的 CPU 数据表中，可以找到关于定时器和计数器的允许编号和关于最大可用的块的数目的更多信息。"

A.3.4.3 参数类型 POINTER 的格式

下图给出了存储在每个字节中的数据类型。



参数类型 POINTER 存储下列信息：

- DB 编号(或 0，如果数据没有存储在 DB 中)
- CPU 中的存储区域(下表给出了参数类型 POINTER 存储器区的十六进制代码)

十六进制代码	存储区域	描述
b#16#81	I	输入区域
b#16#82	Q	输出区域
b#16#83	M	位存储区域
b#16#84	DB	数据块
b#16#85	DI	背景数据块
b#16#86	L	本地的数据(L 堆栈)
b#16#87	V	先前的本地数据

- 数据的地址(格式为字节.位)

  STEP 7 提供指针格式：p#memory_area byte.bit_address.(如果形式参数被声明为参数类型 POINTER，只需要指出存储区域和地址。STEP 7 将自动地重定输入指针的格式。)下面的实例说明如何为以 M50.0 开始的数据输入参数类型 POINTER：

- P#M50.0
- M50.0 (如果形式参数声明为 POINTER)。

A.3.4.4 使用参数类型 POINTER

指针用于指向地址。这种寻址方式的优点是可以在程序处理期间动态地修改语句的地址。

存储器间接寻址的指针

使用存储器间接寻址的程序语句由指令、地址标识符和偏移量组成(偏移量必须在方括号内给出)。

双字格式指针的实例：

L	P#8.7	装载指针的值到累加器 1。
T	MD2	传送指针到 MD2。
A	I [MD2]	询问输入位 I 8.7 的信号状态，并
=	Q [MD2]	将信号状态分配给输出位 Q 8.7。

区域内部和区域交叉寻址的指针

使用这些寻址方式的程序语句包含指令和下列部分：地址标识符、地址寄存器标识符、偏移量。

地址寄存器(AR1/2)和偏移量必须在方括号内一起指定。

区域内部寻址的实例

指针不包含存储器区的指示：

L	P#8.7	装载指针的值到累加器 1。
LAR1		将指针从累加器 1 装载到 AR1。
A	I [AR1, P#0.0]	询问输入位 I 8.7 的信号状态，并
=	Q [AR1, P#1.1]	将信号状态分配给输出位 Q 10.0。

偏移量 0.0 没有影响。输出 10.0 由 8.7(AR1)加上偏移量 1.1 计算出来。结果是 10.0 而不是 9.8，参见指针格式。

区域交叉寻址的实例

在区域交叉寻址中，存储区域在指针中指出(在实例 I 和 Q 中)。

L	P# I8.7	将指针的值和区域标识装载到累加器 1 中。
LAR1		装入存储器区 I 和地址 8.7 到 AR1 中。
L	P# Q8.7	将指针的值和区域标识装载到累加器 1 中。
LAR2		装入存储器区 Q 和地址 8.7 到 AR2 中。
A	[AR1, P#0.0]	询问输入位 I 8.7 的信号状态，
=	[AR2, P#1.1]	将信号状态分配给输出位 Q 10.0。

偏移量 0.0 没有影响。输出 10.0 由 8.7 (AR2)加上偏移量 1.1 计算出来。结果是 10.0 而不是 9.8，参见指针格式。

A.3.4.5 改变指针的块

使用采样块 FC3“路由指针”能够改变指针的位或字节地址。当调用 FC 时，要改变的指针传送到变量“指针”(可以使用以双字格式的区域内部和区域交叉指针)。

使用参数“位-字节”可以改变指针的位或字节地址(0：位地址，1：字节地址)。变量“Inc_Value”(整数格式)指定增加或减少地址内容的数字。也可以指定负数以减少地址。

改变位地址，字节地址也有所改变(当减少时也是)，例如：

- P#M 5.3, Bit_Byte = 0, Inc_Value = 6 => P#M 6.1 或
- P#M 5.3, Bit_Byte = 0, Inc_Value = -6 => P#M 4.5。

指针的区域信息不受功能影响。

FC 截取指针的上溢/下溢。在此情况下，指针不改变，输出变量“RET_VAL”(可能的错误处理)设置为“1”(直到下一个 FC3 的更正处理)。此处是个例子：

- 1. 位地址已选择，Inc_Value >7，或<-7
- 2. 位或字节地址已选择，改变可能导致“负的”字节地址
- 3. 位或字节地址已选择，改变可能导致非法的大的字节地址。

改变指针的 STL 中采样块

```
FUNCTION FC 3:
TITLE =Routing Pointers
//FC3 可以用于改变指针。
AUTHOR :AUT1CS1
FAMILY :INDADDR
NAME :ADDRPOINT
VERSION : 0.0

VAR_INPUT
    Bit_Byte :BOOL ; //0：位地址，1：字节地址
    Inc_Value :INT ; //增加(如果值负 => 减少/如果值正 => 增加)
END_VAR

VAR_IN_OUT
    Pointer : DWORD ; //要改变的指针
END_VAR
VAR_TEMP
    Inc_Value1 :INT ; //中间值增加
    Pointer1 :DWORD ; //中间值指针
    Int_Value :DWORD ; //辅助变量
END_VAR
BEGIN
NETWORK
TITLE =
//块截取改变指针区域信息的改变
//或自动地导致“负的”指针的改变。
    SET     ; //设置 RLO 为 1
    R       #RET_VAL; //复位上溢
    L       #Pointer; //提供临时值
    T       #Pointer1; //中间值指针
    L       #Inc_Value; //提供临时值
    T       #Inc_Value1; //中间值增加
    A       #Bit_Byte; //If =1，字节地址指令
    JC      Byte; //跳转到字节地址计算
    L       7; //如果增加值> 7，
    L       #Inc_Value1;
    <I      ;
    S       #RET_VAL; //然后设置 RET_VAL
    JC      End; //跳转到 End
    L       -7; //如果增加值 < -7，
    <I      ;
    S       #RET_VAL; //然后设置 RET_VAL
    JC      End; //跳转到 End
```

```
      A      L      1.3; //如果该值的位 4 = 1(Inc_Value 为负)
      JC     neg; //然后跳转到位地址减少
      L      #Pointer1; //装载指针地址信息
      L      #Inc_Value1; //加上增加量
      +D     ;
      JU     test; //跳转到负的结果测试
neg:         L      #Pointer1; //装载指针地址信息
      L      #Inc_Value1; //装载增加量
      NEGI   ; //取反负值,
      -D     ; //减去该值
      JU     test; //并跳转到测试
Byte:        L      0; //开始字节地址的改变
      L      #Inc_Value1; //如果增加量 >=0,那么
      <I     ;
      JC     pos; //跳转到增加,否则
      L      #Pointer1; //装载指针地址信息,
      L      #Inc_Value1; //装载增加量,
      NEGI   ; //取反负值,
      SLD    3; //向左移位增加量 3,
      -D     ; //减去该值,
      JU     test; //并跳转到测试
pos:         SLD    3; //向左移位增加量 3
      L      #Pointer1; //装载指针地址信息
      +D     ; //加上增加量
test:        T      #Int_Value; //传送计算的结果到 Int_Value
      A      L      7.3; //如果无效的字节地址(太长或
      S      #RET_VAL; //负的),然后设置 RET_VAL
      JC     End; //并跳转到结束,
      L      #Int_Value; //否则传送结果
      T      #Pointer; //到指针
End:         NOP    0;
END_FUNCTION
```

A.3.4.6 参数类型 ANY 的格式

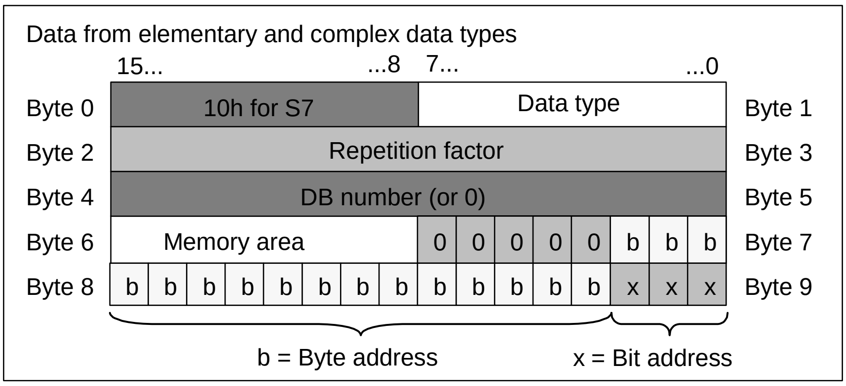
STEP 7 以 10 个字节存储参数类型 ANY。当构造类型为 ANY 的参数时，必须确保所有 10 个字节都被占用，因为调用块估计参数整个内容的值。例如，如果以字节 4 指定 DB 编号，也必须以字节 6 明确地指定存储器区。

STEP 7 管理基本和复杂数据类型的数据与参数类型的数据不同。

数据类型的 ANY 格式

对于基本和复杂数据类型，STEP 7 存储下列数据：

- 数据类型
- 重复因子
- DB 编号
- 信息存储的存储区域
- 数据的起始地址



重复因子识别由参数类型 ANY 传送的指示数据类型的数量。这意味着可以指定数据区，也可以和参数类型 ANY 结合使用数组和结构。STEP 7 将数组和结构识别为数据类型的编号(借助重复因子)。例如，如果要传送 10 个字，必须为重复因子输入数值 10，并且必须为数据类型输入数值 04。

地址以格式 Byte.Bit 存储，此处字节寻址存储在字节 7 的位 0 - 2，节字 8 的位 0 - 7，节字 9 的位 3 - 7。位地址存储在字节 9 的位 0 - 2。

对于类型 NIL 的空指针，所有来自字节 1 的字节设置为 0。

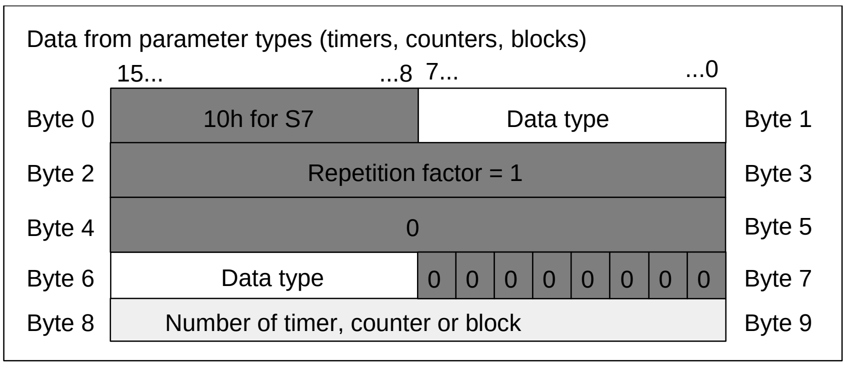
下表给出参数类型 ANY 的数据类型或存储区域的编码。

数据类型的编码		
十六进制代码	数据类型	描述
b#16#00	NIL	空指针
b#16#01	BOOL	位
b#16#02	BYTE	字节(8 位)
b#16#03	CHAR	字符(8 位)
b#16#04	WORD	字(16 位)
b#16#05	INT	整数(16 位)
B#16#06	DWORD	字(32 位)
b#16#07	DINT	双整数(32 位)
b#16#08	REAL	浮点数(32 位)
b#16#09	DATE	日期
b#16#0A	TIME_OF_DAY (TOD)	时间
b#16#0B	TIME	时间
b#16#0C	S5TIME	数据类型 S5TIME
b#16#0E	DATE_AND_TIME (DT)	日期和时间(64 位)
b#16#13	STRING	字符串

存储器区的编码		
十六进制代码	区域	描述
b#16#81	I	输入区域
b#16#82	Q	输出区域
b#16#83	M	位存储区域
b#16#84	DB	数据块
b#16#85	DI	背景数据块
b#16#86	L	本地的数据(L 堆栈)
b#16#87	V	先前的本地数据

参数类型的 ANY 格式

对于参数类型，STEP 7 存储数据类型和参数的地址。重复因子始终是 1。字节 4、5 和 7 始终是 0。字节 8 和 9 指示定时器、计数器或块的编号。



下表为参数类型显示参数类型 ANY 的数据类型编码。

十六进制代码	数据类型	描述
b#16#17	BLOCK_FB	FB 编号
b#16#18	BLOCK_FC	FC 编号
b#16#19	BLOCK_DB	DB 编号
b#16#1A	BLOCK_SDB	SDB 编号
b#16#1C	COUNTER	计数器编号
b#16#1D	TIMER	定时器编号

A.3.4.7 使用参数类型 ANY

可以为适合于任何数据类型的实际参数的块定义形式参数。当调用块是未知或可以改变时(和当允许任何数据类型时)，已提供了实际参数的数据类型时，这尤其有用。在块的变量声明中，可以声明参数为数据类型 ANY。然后可以在 STEP 7 中分配任何数据类型的实际参数。

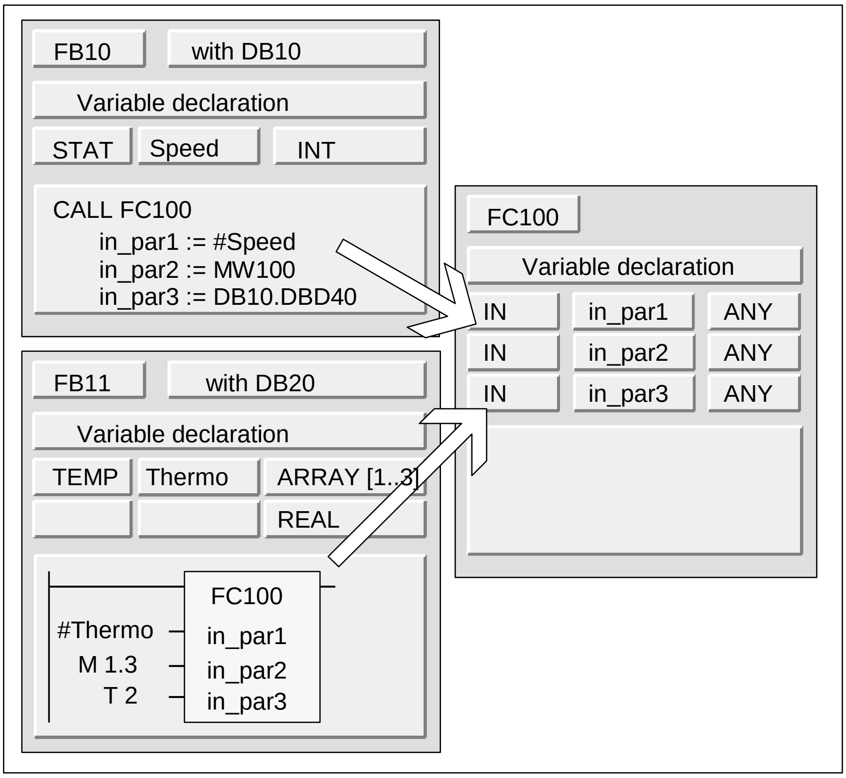
STEP 7 为 ANY 数据类型的变量分配存储器的 80 个位。如果分配实际参数给此形式参数，STEP 7 在 80 个位中编码起始地址、数据类型和实际参数的长度。调用块为 ANY 参数分析保存数据的 80 个位，并获取进一步处理所需的信息。

分配实际参数给 ANY 参数

如果为参数声明数据类型 ANY，可以分配任何数据类型的实际参数给形式参数。在 STEP 7 中，可以指定下列数据类型为实际参数：

- 基本数据类型：指定实际参数的绝对地址或符号名称。
- 复杂数据类型：指定复杂数据类型的数据符号名称(例如，数组和结构)。
- 定时器、计数器和块：指定编号(例如，T1、C20 或 FB6)。

下图说明数据如何传送到具有 ANY 数据类型参数的 FC。



在此实例中，FC100 具有三个参数(*in_par1*、*in_par2 和 in_par3*)，声明为 ANY 数据类型。

- 当 FB10 调用 FC100 时，FB10 传送一个整数(静态变量 speed)、一个字(MW100)和一个双字到 DB10(DB10.DBD40)。
- 当 FB11 调用 FC100 时，FB11 传送一个实数数组(临时变量“Thermo”)、一个布尔值(M 1.3)和一个定时器(T2)。

为 ANY 参数指定数据区

不仅可以分配各个地址(例如，MW100)给 ANY 参数，而且可以指定数据区。如果希望指定数据区为实际参数，使用下列常数格式指定要传送的数据总量：

p#　*区域标识符 Byte.Bit　数据类型　重复因子*

对于*数据类型*元素，可以为常数指定所有基本数据类型和数据类型 DATE_AND_TIME 的格式。如果数据类型不是 BOOL，必须指定位地址 0(x.0)。下表为指定要传送的存储区域给 ANY 参数的格式的实例。

实际参数	描述
p# M 50.0 BYTE 10	指定字节存储区域中的 10 个字节： MB50 到 MB59。
p# DB10.DBX5.0 S5TIME 3	指定数据类型 S5TIME 的数据的 3 个单元，它们位于 DB10： DB 字节 5 到 DB 字节 10。
p# Q 10.0 BOOL 4	在输出区指定 4 个位 Q 10.0 到 Q 10.3.

使用参数类型 ANY 的实例

下列实例给出了如何使用参数类型 ANY 和系统功能 SFC20 BLKMOV 复制 10 个字节的存储区域。

```
STL                          解释
FUNCTION FC10:VOID
VAR_TEMP
  Source  : ANY;
  Target  : ANY;
END_VAR
BEGIN
LAR1    P#Source;            在 AR1 中装载 ANY 指针的起始地址。

L  B#16#10;                  装载语法标识符并将它
T  LB[AR1,P#0.0];            传送给 ANY 指针。

L  B#16#02;                  装载数据类型字节并
T  LB[AR1,P#1.0];            将它传送到 ANY 指针。

L  10;                       装载 10 个字节并将它们
T  LW[AR1,P#2.0];            传送到 ANY 指针。

L  22;                       源是 DB22、DBB11
T  LW[AR1,P#4.0];
L  P#DBX11.0;
T  LD[AR1,P#6.0];

LAR1    P#Target;            在 AR1 中装载 ANY 指针的起始地址。

L  B#16#10;                  装载语法标识符并将它
T  LB[AR1,P#0.0];            传送给 ANY 指针。

L  B#16#02;                  装载数据类型字节并
T  LB[AR1,P#1.0];            将它传送到 ANY 指针。

L  10;                       装载 10 个字节并将它们
T  LW[AR1,P#2.0];            传送到 ANY 指针。

L  33;                       目标是 DB33、DBB202
T  LW[AR1,P#4.0];
L  P#DBX202.0;
T  LD[AR1,P#6.0];

CALL SFC 20(                 调用系统功能 BLKMOV
    SRCBLK  := Source,
    RET_VAL := MW 12,        求 BR 位和 MW12 的值
    DSTBLK  := Target
     );
END_FUNCTION
```

A.3.4.8 分配数据类型给逻辑块的本地数据

对于 STEP 7，在变量声明中，分配给块的本地数据的数据类型(基本和复杂数据类型和参数类型)受限制。

OB 的本地数据的有效数据类型

下表说明了为 OB 声明本地数据的限定(-) 。因为不能调用 OB，OB 不能有参数(输入、输出或输入/输出)。因为 OB 没有背景数据块，不能为 OB 声明任何静态变量。OB 临时变量的数据类型可以是基本或复杂数据类型以及数据类型 ANY。

有效的分配由●符号显示。

声明类型	基本数据类型	复杂数据类型	参数类型	参数类型	参数类型	参数类型	参数类型
			TIMER	COUNTER	BLOCK	POINTER	ANY
输入	—	—	—	—	—	—	—
输出	—	—	—	—	—	—	—
输入/输出	—	—	—	—	—	—	—
静态	—	—	—	—	—	—	—
临时	● (1)	● (1)	—	—	—	—	● (1)
(1) 位于 OB 的 L 堆栈。							

FB 的本地数据的有效数据类型

下表说明了为 FB 声明本地数据的限定(-)。至于背景数据块，在声明 FB 本地数据时限制较少。当声明输入参数时，没有什么限制；对于输出参数，不能声明任何参数类型，对于输入/输出参数仅允许参数类型 POINTER 和 ANY。可以声明临时变量为 ANY 数据类型。其它所有参数类型是非法的。

有效的分配由●符号显示。

声明类型	基本数据类型	复杂数据类型	参数类型	参数类型	参数类型	参数类型	参数类型
			TIMER	COUNTER	BLOCK	POINTER	ANY
输入	●	●	●	●	●	●	●
输出	●	●	—	—	—	—	—
输入/输出	●	● (1)(3)	—	—	—	●	●
静态	●	●	—	—	—	—	—
临时	● (2)	● (2)	—	—	—	—	● (2)
(1) 在背景数据块中作为引用(48 位指针)存储。 (2) 位于 FB 的 L 堆栈。 (3) STRING 只能以缺省长度定义。							

FC 的本地数据的有效数据类型

下表说明了为 FC 声明本地数据的限定(-)。因为 FC 没有背景数据块，也没有静态变量。对于 FC 的输入、输出和输入/输出参数，只允许参数类型 POINTER 和 ANY。也可以声明 ANY 参数类型的临时变量。

有效的分配由●符号显示。

声明类型	基本数据类型	复杂数据类型	参数类型	参数类型	参数类型	参数类型	参数类型
			TIMER	COUNTER	BLOCK	POINTER	ANY
输入	●	● (2)	●	●	●	●	●
输出	●	● (2)	—	—	—	●	●
输入/输出	●	● (2)	—	—	—	●	●
临时	● (1)	● (1)	—	—	—	—	● (1)

(1) 位于 FC 的 L 堆栈。

(2) STRING 只能以缺省长度定义。

A.3.4.9 在传送参数时允许的数据类型

在块之间传送参数的规则

当分配实际参数给形式参数时，可以指定绝对地址、符号名称或常数。STEP 7 限制不同参数的有效分配。例如，输出和输入/输出参数不能被分配常数值(因为输出或输入/输出参数的目的是改变其值)。这些限定尤其适用于具有复杂数据类型的参数，这些参数既不能分配绝对地址也不能分配常数。

下表说明涉及分配给形式参数的实际参数数据类型的限制(-)。

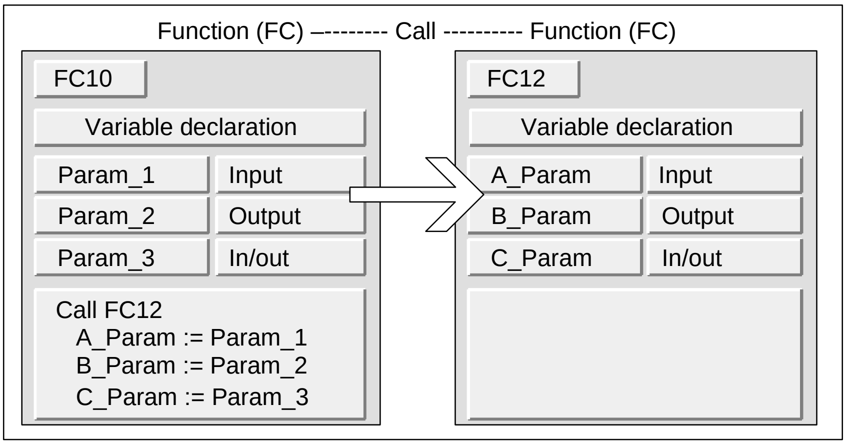
有效的分配由●符号显示。

基本数据类型				
声明类型	绝对地址	符号名称 (在符号表中)	临时本地符号	常数
输入	●	●	●	●
输出	●	●	●	—
输入/输出	●	●	●	—

复杂数据类型				
声明类型	绝对地址	DB 元素的符号名称 (在符号表中)	临时本地符号	常数
输入	—	●	●	—
输出	—	●	●	—
输入/输出	—	●	●	—

通过函数进行函数调用的有效数据类型

可以将调用 FC 的形式参数分配给被调用 FC 的形式参数。下图说明了 FC10 的形式参数作为实际参数分配给 FC12 的形式参数。



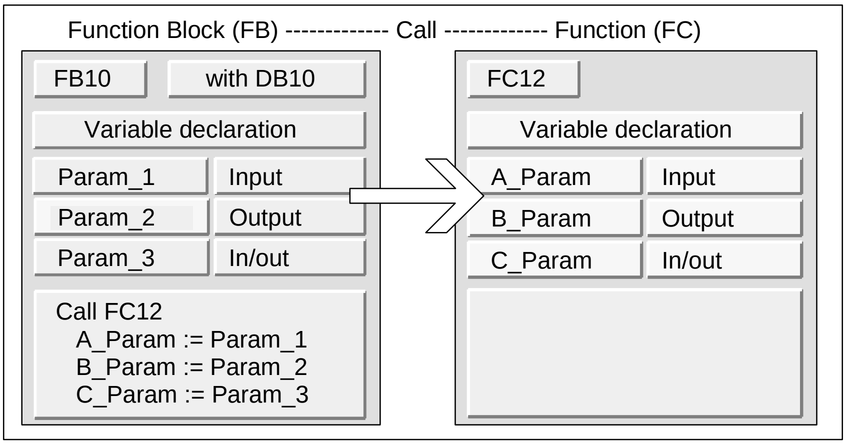
STEP 7 限制将 FC 形式参数作为实际参数分配给不同 FC 的形式参数。例如，不能将具有复杂数据类型的参数或参数类型分配为实际参数。

下表给出了当一个 FC 调用另一个 FC 时允许的数据类型(●)。

声明类型	基本数据类型	复杂数据类型	参数类型	参数类型	参数类型	参数类型	参数类型
			TIMER	COUNTER	BLOCK	POINTER	ANY
输入 → 输入	●	—	—	—	—	—	—
输入 → 输出	—	—	—	—	—	—	—
输入 → 输入/输出	—	—	—	—	—	—	—
输出 → 输入	—	—	—	—	—	—	—
输出 → 输出	●	—	—	—	—	—	—
输出 → 输入/输出	—	—	—	—	—	—	—
输入/输出 → 输入	●	—	—	—	—	—	—
输入/输出 → 输出	●	—	—	—	—	—	—
输入/输出 → 输入/输出	●	—	—	—	—	—	—

通过功能块调用函数的有效数据类型

可以将调用 FB 的形式参数分配给被调用 FC 的形式参数。下图说明了将 FB10 的形式参数作为实际参数分配给 FC12 的形式参数。

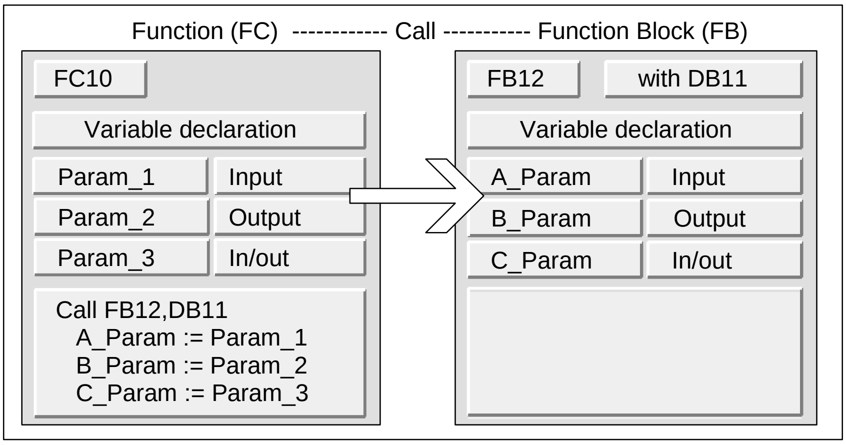


STEP 7 限制将 FB 的形式参数分配给 FC 的形式参数。例如，不能分配参数类型的参数作为实际参数。下表给出了 FB 调用 FC 时允许的数据类型(●)。

声明类型	基本数据类型	复杂数据类型	参数类型	参数类型	参数类型	参数类型	参数类型
			TIMER	COUNTER	BLOCK	POINTER	ANY
输入 → 输入	●	●	—	—	—	—	—
输入 → 输出	—	—	—	—	—	—	—
输入 → 输入/输出	—	—	—	—	—	—	—
输出 → 输入	—	—	—	—	—	—	—
输出 → 输出	●	●	—	—	—	—	—
输出 → 输入/输出	—	—	—	—	—	—	—
输入/输出 → 输入	●	—	—	—	—	—	—
输入/输出 → 输出	●	—	—	—	—	—	—
输入/输出 → 输入/输出	●	—	—	—	—	—	—

通过函数调用功能块的有效数据类型

可以将调用 FC 的形式参数分配给被调用 FB 的形式参数。下图说明了 FC10 的形式参数作为实际参数分配给 FB12 的形式参数。



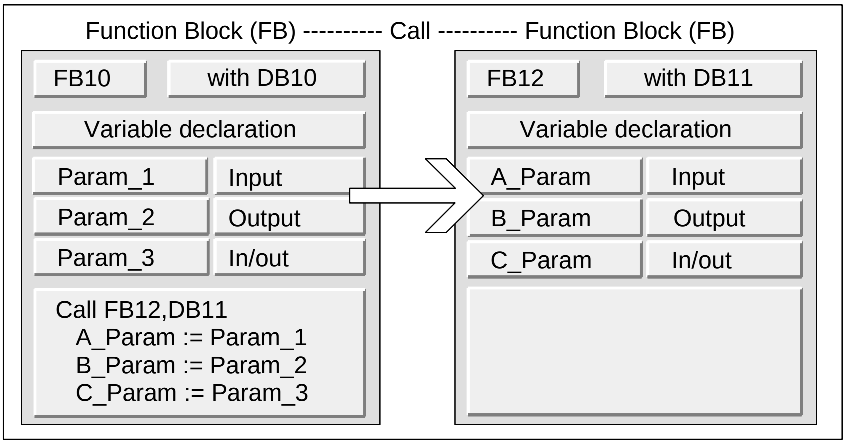
STEP 7 限制将 FC 的形式参数分配给 FB 的形式参数。例如，不能分配具有复杂数据类型的参数作为实际参数。然而，可以分配参数类型 TIMER、COUNTER 或 BLOCK 的输入参数给被调用 FB 的输入参数。

下表给出了 FC 调用 FB 时允许的数据类型(●)。

声明类型	基本数据类型	复杂数据类型	参数类型	参数类型	参数类型	参数类型	参数类型
			TIMER	COUNTER	BLOCK	POINTER	ANY
输入 → 输入	●	—	●	●	●	—	—
输入 → 输出	—	—	—	—	—	—	—
输入 → 输入/输出	—	—	—	—	—	—	—
输出 → 输入	—	—	—	—	—	—	—
输出 → 输出	●	—	—	—	—	—	—
输出 → 输入/输出	—	—	—	—	—	—	—
输入/输出 → 输入	●	—	—	—	—	—	—
输入/输出 → 输出	●	—	—	—	—	—	—
输入/输出 → 输入/输出	●	—	—	—	—	—	—

通过功能块调用功能块的有效数据类型

可以将调用 FB 的形式参数分配给被调用 FB 的形式参数。下图说明了 FB10 的形式参数作为实际参数分配给 FB12 的形式参数。



STEP 7 限制分配 FB 的形式参数给另一个 FB 的形式参数。例如，不能分配具有复杂数据类型的输入和输出参数作为被调用 FB 的输入和输出参数的实际参数。然而，可以分配参数类型 TIMER、COUNTER 或 BLOCK 的输入参数给被调用 FB 的输入参数。

下表给出了 FB 调用另一个 FB 时允许的数据类型(●)。

声明类型	基本数据类型	复杂数据类型	参数类型	参数类型	参数类型	参数类型	参数类型
			TIMER	COUNTER	BLOCK	POINTER	ANY
输入 → 输入	●	●	●	●	●	—	—
输入 → 输出	—	—	—	—	—	—	—
输入 → 输入/输出	—	—	—	—	—	—	—
输出 → 输入	—	—	—	—	—	—	—
输出 → 输出	●	●	—	—	—	—	—
输出 → 输入/输出	—	—	—	—	—	—	—
输入/输出 → 输入	●	—	—	—	—	—	—
输入/输出 → 输出	●	—	—	—	—	—	—
输入/输出 → 输入/输出	●	—	—	—	—	—	—

A.3.4.10 传送到功能块的 IN_OUT 参数

当复杂数据类型传送给功能块(FB)的 IN_OUT 参数时，同时也传送变量的地址(由引用调用)。

当基本数据类型传送给功能块的 IN_OUT 参数时，在功能块开始执行之前，将值复制到背景数据块中，在功能块结束后从背景数据块中将值复制出来。

这意味着基本数据类型的 IN_OUT 变量可以用值初始化。

然而，不可能在 IN_OUT 变量的位置指定常数作为在调用中的实际参数，因为不能写入常数。

数据类型 STRUCT 或 ARRAY 的变量无法初始化，因为在此情况下仅有一个地址位于背景数据块中。

A.4 使用较早的项目

A.4.1 转换版本 1 的项目

可以重复使用通过 STEP 7 版本 1 创建的项目。为此，必须将版本 1 项目转换为版本 2 项目。

版本 1 项目的下列组件保持不变：

- 程序的项目结构
- 块
- STL 源文件
- 符号表

硬件配置不转换。可以将项目中包含的程序组件复制到其它项目中。也可以增加一个站到新项目，并且为它组态和分配参数。

一旦转换到版本 2，就可以在对话框中决定，现在是否希望将版本 2 项目转换为当前 STEP 7 版本项目。

注意

就属性而言，单个块保持为版本 1 块。在版本 1 中生成的代码不会改变，因此块不能结合多重背景一起使用。

如果希望在所转换的块中声明多重背景，请先使用“LAD/STL/FBD：程序块”应用程序，从所转换的块生成 STL 源文件，然后将它们编译回块中。

编程多重背景是 STEP 7 版本 2 的新特性，可用于创建功能块(FB)。如果希望在版本 2 项目中以相同的方法继续使用以版本 1 创建的功能块，则不需要转换它们。

步骤

要转换版本 1 项目，步骤如下：

1. 选择菜单命令文件 > 打开版本 1 项目。
2. 在出现的对话框中，选择要在版本 2 中使用的版本 1 项目。可通过项目扩展名 *.s7a 识别版本 1 项目。
3. 然后，在下一个对话框中，输入希望转换的版本 1 项目的新项目名称。

A.4.2 转换版本 2 的项目

在 STEP 7 中，还可以使用菜单命令**文件 > 打开**来打开版本 2 项目。

可以使用菜单命令**文件 > 另存为**和选项“保存之前重新排列”，将版本 2 项目/库转换(移植)到 STEP 7 当前版本。可将版本 2 的项目/库转换(移植)到当前的 STEP 7。

然后，项目被另存为当前 STEP 7 版本的项目。可以编辑来自 STEP 7 旧版本中的项目和库，保留其格式，并在“项目另存为”对话框中选择 STEP 7 旧版本作为文件类型。例如，要编辑版本为 STEP 7 版本 2.1 的对象，在此处选择“项目 2.x”或“库 2.x”(从版本 5.1 开始，不可以另存为版本 2。也可以参见编辑版本 2 的项目和库)。

文件类型标志

	STEP 7 V3	从 **STEP 7 V4** 起
当前版本的文件类型	Project3.x Library3.x	Project Library
旧版本的文件类型	Project2.x Library2.x	Project2.x Library2.x

这意味着只能访问 STEP 7 旧版本的功能范围。不过，仍可以继续管理 STEP 7 旧版本的项目和库。

注意

从版本 3 升级至版本 4 及更高版本只涉及名称的改变：格式保持相同。因此，在 STEP 7 V4 中没有“Project3.x”文件类型。

步骤

要将版本 2 项目转换为当前 STEP 7 版本的格式，可如下操作：

1. 在“文件”菜单中执行“另存为”命令，选中“在保存前重新排列”选项。
2. 在“项目另存为”对话框中，选择“项目”文件类型，并点击“保存”按钮。

要将版本 2 项目转换为当前 STEP 7 版本，同时保留它们的格式，过程如下：

1. 如有必要，执行上述的步骤 1。
2. 在“项目另存为”对话框中，选择 STEP 7 旧版本的文件类型，并点击“保存”按钮。

A.4.3 关于具有 GD 通讯的 STEP 7 V.2.1 项目的注意事项

- 如果希望将具有全局数据的项目从 STEP 7 V2.1 转换到 STEP 7 V5，必须先在 STEP 7 V2.1 项目中用 STEP 7 V5.0 打开全局数据表。以前组态的通讯数据通过 GD 通讯自动转换为新结构。
- 当归档 STEP 7 V2.1 项目时，如果项目中含有名称长度大于八个字符的文件，则旧程序(ARJ、PKZIP...)可能发布错误消息。如果编辑 STEP 7 V2.1 项目中的 MPI 程序段时，所用标识符的长度大于 8 个字符，此消息也会出现。在具有全局数据的 STEP 7 V2.1 项目中，首次开始组态全局数据通讯之前，请为 MPI 程序段编辑一个最大长度为八个字符的名称。
- 如果希望重命名 STEP 7 V2.1 项目，就必须通过重新选择合适的 CPU，重新分配 GD 表中的栏(CPU)标题。如果恢复旧的项目名称，该分配会再次显示。

A.4.4 具有丢失或故障 GSD 文件的 DP 从站

如果用 STEP 7 版本 5.1 处理更旧的站组态，在极少的情况下，DP 从站的 GSD 文件会丢失或不能编译(例如，由于 GSD 文件中的语法错误)。

在这种情况下，STEP 7 生成代表组态从站的“占位”从站。例如，在站下载到编程设备后，或更旧的项目已打开和进一步处理后。此“占位”从站只能在有限的程度内处理。不能改变从站的结构(DP 标识符)和从站的参数。不过，可以重新下载到站。从站的原始组态仍保持。也可以删除整个 DP 从站。

重新组态和设置参数到 DP 从站

如果希望重新组态或重新分配参数到 DP 从站，必须向制造商要求此 DP 从站的最新 GSD 文件，并通过菜单命令**选项 > 安装新** GSD 使之可用。

在安装了正确的 GSD 文件后，它用于表示 DP 从站。DP 从站包含其数据，并可以再次全面处理。

A.5 实例程序

A.5.1 实例项目和实例程序

STEP 7 安装光盘包含了大量实例项目，列举如下。您可在 SIMATIC 管理器中的“打开”对话框中找到这些实例项目(“实例项目”标签)。当安装有选项包时，也可添加其它的实例项目。关于这些实例项目的有关信息，请参见选项包的相关文档。

实例和实例项目	**在 CD 中已包含**	在本文档中描述	**OB1 中的描述**
“ZEn01_01_STEP7_*”到“ZEn01_06_STEP7_*”项目(入门与练习)	•	单独的手册	•
“ZEn01_11_STEP7_DezP”项目(实例 PROFIBUS DP 组态)	•	–	–
“ZEn01_08_STEP7_Blending”项目(工业混合过程)	•	•	–
“ZEn01_09_STEP7_Zebra”项目(斑马线/人行过道的交通信号控制)	•		•
“Zen01_10_STEP7_COM_SFB”项目(两个 S7-400 CPU 之间的数据交换)	•		•
“ZXX01_14_HSystem_S7400H”项目(容错系统项目入门)	•	单独的手册	•
“ZXX01_15_HSystem_RED_IO 项目(具有冗余 I/O 设备的容错系统项目入门)	•	单独的手册	•
“Zen01_11_STEP7_COM_SFC1”和“Zen01_12_STEP7_COM_SFC2”项目(使用未组态连接的通讯 SFC 进行的数据交换)	•		•
项目“ZEn01_13_STEP7_PID-Temp”(用于温度控制器 FB 58 和 FB 59 的实例)	•		•
处理时间中断的实例		•	
处理中断的实例		•	
对同步错误进行掩码和消除掩码的实例		•	
禁止和激活中断和异步错误的实例		•	
对中断和异步错误进行延迟处理的实例		•	

实例的重点不在于教授特定的程序设计方式或控制特定过程所需要的专业知识。实例只是旨在说明设计一个程序所必须遵守的步骤。

删除和安装所提供的实例项目

所提供的实例项目可在 SIMATIC 管理器中删除，然后再重新安装。为了安装实例项目，您必须启动 STEP 7 V5.0 安装程序。也可选择日后安装实例项目。 提供的实例项目的副本以及使用菜单命令“另存为”自行创建的实例项目只能作为用户项目保存。

注意

当安装 STEP 7 时，除非指定不安装，否则将复制所提供的实例项目。如果您已经编辑了所提供的实例项目，在重新安装 STEP 7 时，这些修改过的项目将被原始项目覆盖。

因此，在修改前，您应复制所提供的实例项目，然后只对副本进行编辑。

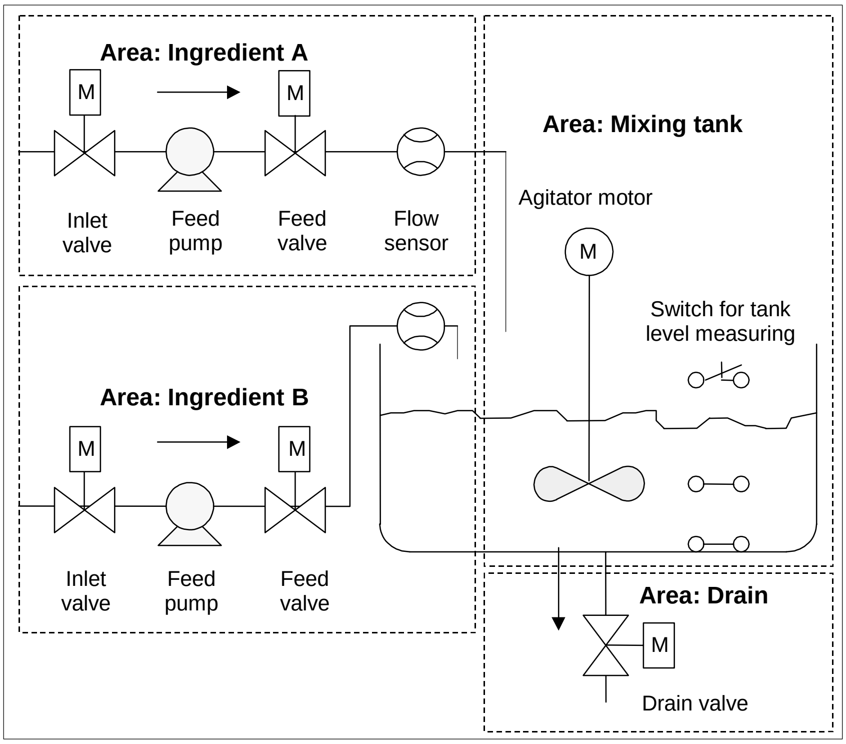
A.5.2 工业混合过程的示例程序

A.5.2.1 工业混合过程的实例程序

示例程序使用在手册第一部分中读到过的关于控制工业混合过程的信息。

任务

搅拌机在混合罐中将两种配料(配料 A 和配料 B)混和在一起。混合好的产品通过排料阀从混合罐中排出。下图显示了示例过程的框图。



过程各部分描述

手册第 1 部分描述了如何将示例过程分割为功能区域和单个任务。各区域说明如下。

用于配料 A 和 B 的区域：

- 每种配料管道都装备有入口阀、进料阀和进料泵。
- 入口管还装有流量传感器。
- 当混合罐液位传感器显示混合罐已满时，打开进料泵必须联锁。
- 当排料阀打开时，进料泵的起动必须联锁。

- 在启动进料泵后的最初 1 秒，入口和进料阀必须打开。
- 在进料泵停止后(来自流量传感器的信号)，必须立即关闭阀，以防止配料从泵泄漏。
- 进料泵的起动与时间监视功能相关联。换句话说，在泵启动后 7 秒内，流量传感器必须报告流量。
- 如果在进料泵运行时，流量传感器不再发出流量信号，则必须尽快关闭进料泵。
- 必须对进料泵的启动次数进行计数(维护时间间隔)。

混合罐区域：

- 当混合罐液位传感器指示“液位低于最小值”或排料阀打开时，必须联锁搅拌器电机的起动。
- 在达到额定速度后，搅拌器电机发送响应信号。如果电机起动后 10 秒内没有收到该信号，则必须关闭电机。
- 必须计数搅拌机电机启动的次数(维护时间间隔)。
- 必须在混合罐中安装三个传感器：
    - 混合罐满：常闭触点。当达到搅拌罐最高液位时，触点断开。
    - 混合罐中的液位高于最小值：常开触点。如果达到最低液位，触点闭合。
    - 混合罐不空：常开触点。如果混合罐不是空的，则触点闭合。

排料区：

- 混合罐的排料由电磁阀控制。
- 电磁阀由操作员控制，但最迟当产生“混合罐空”信号时，必须再次关闭电磁阀。
- 下列情况下，打开排料阀是联锁的
    - 搅拌器电机正在运行
    - 混合罐空

操作员站

为了让操作员启动、停止和监视过程，还需要操作员站。操作员站配备下列设备：

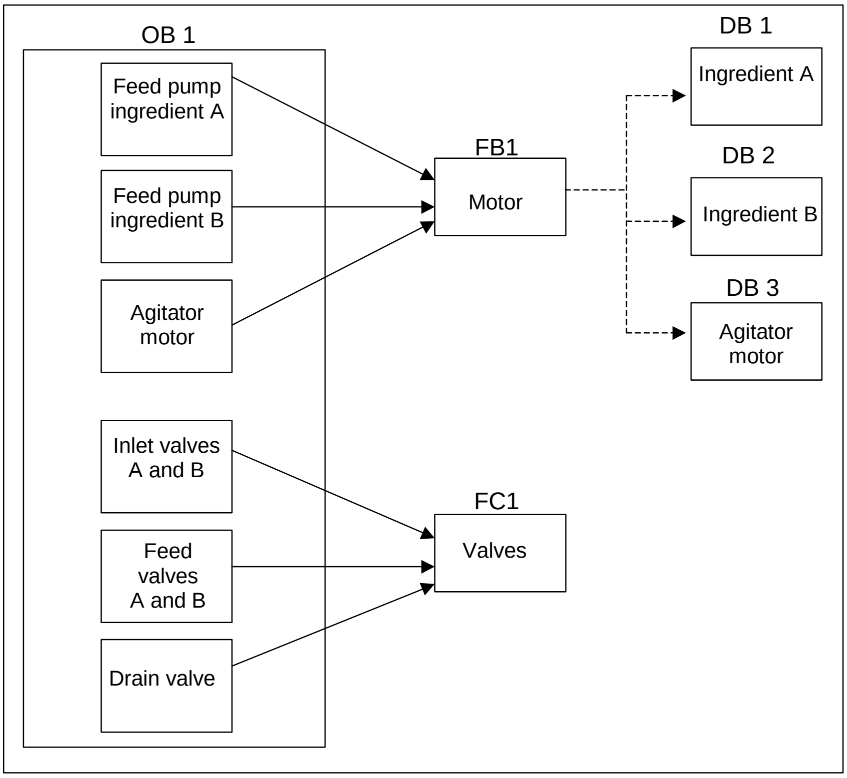
- 用于控制最重要的过程阶段的开关。使用“复位维护显示”开关，可以在维护时关闭电机的维护指示灯，并将相应的维护时间间隔计数器复位为 0。
- 指示过程状态的指示灯。
- 紧急停车开关。

A.5.2.2 定义逻辑块

通过分配用户程序给各个块及建立块调用层次，建立程序结构。

块调用层次

下图显示在结构化程序中，被调用的块的层次。



- OB1：CPU 操作系统接口，包含主程序。在 OB1 中调用块 FB1 和 FC1，并传送过程控制所需要的特定参数。
- FB1：配料 A 的进料泵、配料 B 的进料泵和搅拌机电机可以由一个单独的功能块控制，因为它们的要求(开、关、计数应用等)是相同的。
- 背景数据块 1-3：用于控制配料 A、配料 B 的进料泵和搅拌器电机的实际参数和静态数据是不同的，因此存储在三个与 FB1 关联的背景数据块中。
- FC1：配料 A 和 B 的入口阀、进料阀以及排料阀也使用共用逻辑块。由于只是对“打开和关闭”功能是必须进行编程的，所以使用单个的 FC 已足够。

A.5.2.3 分配符号名

定义符号名

符号在示例程序中使用，它们必须用 STEP 7 在符号表中定义。下表给出了符号名和程序元素所使用的绝对地址。

进料泵、搅拌机电机和入口阀的符号地址			
符号名	地址	数据类型	描述
Feed_pump_A_start	I0.0	BOOL	启动配料 A 的进料泵
Feed_pump_A_stop	I0.1	BOOL	停止配料 A 的进料泵
Flow_A	I0.2	BOOL	配料 A 流量
Inlet_valve_A	Q4.0	BOOL	起动配料 A 的入口阀
Feed_valve_A	Q4.1	BOOL	起动配料 A 的进料阀
Feed_pump_A_on	Q4.2	BOOL	“配料 A 进料泵运行”灯
Feed_pump_A_off	Q4.3	BOOL	“配料 A 进料泵不运行”灯
Feed_pump_A	Q4.4	BOOL	起动配料 A 的进料泵
Feed_pump_A_fault	Q4.5	BOOL	“进料泵 A 故障”灯
Feed_pump_A_maint	Q4.6	BOOL	“进料泵 A 维护”灯
Feed_pump_B_start	I0.3	BOOL	启动配料 B 的进料泵
Feed_pump_B_stop	I0.4	BOOL	停止配料 B 的进料泵
Flow_B	I0.5	BOOL	配料 B 的流量
Inlet_valve_B	Q5.0	BOOL	起动配料 A 的入口阀
Feed_valve_B	Q5.1	BOOL	起动配料 B 的进料阀
Feed_pump_B_on	Q5.2	BOOL	“配料 B 进料泵运行”灯
Feed_pump_B_off	Q5.3	BOOL	“配料 B 进料泵不运行”灯
Feed_pump_B	Q5.4	BOOL	起动配料 B 的进料泵
Feed_pump_B_fault	Q5.5	BOOL	“进料泵 B 故障”灯
Feed_pump_B_maint	Q5.6	BOOL	“进料泵 B 维护”灯
Agitator_running	I1.0	BOOL	搅拌机电机的响应信号
Agitator_start	I1.1	BOOL	搅拌机启动按钮
Agitator_stop	I1.2	BOOL	搅拌机停止按钮
Agitator	Q8.0	BOOL	起动搅拌机
Agitator_on	Q8.1	BOOL	“搅拌机运行”灯
Agitator_off	Q8.2	BOOL	“搅拌机不运行”灯
Agitator_fault	Q8.3	BOOL	“搅拌机电机故障”灯
Agitator_maint	Q8.4	BOOL	“搅拌机电机维护”灯

传感器符号地址及显示混合罐的液位			
符号名	地址	数据类型	描述
Tank_below_max	I1.3	BOOL	“混合罐未满”传感器
Tank_above_min	I1.4	BOOL	“混合罐超过最低液位”传感器
Tank_not_empty	I1.5	BOOL	“混合罐不空”传感器
Tank_max_disp	Q9.0	BOOL	“混合罐满”灯
Tank_min_disp	Q9.1	BOOL	“混合罐低于最低液位”灯
Tank_empty_disp	Q9.2	BOOL	“混合罐空”灯

排料阀的符号地址			
符号名	地址	数据类型	描述
Drain_open	I0.6	BOOL	用于打开排料阀的按钮
Drain_closed	I0.7	BOOL	用于关闭排料阀的按钮
Drain	Q9.5	BOOL	起动排料阀
Drain_open_disp	Q9.6	BOOL	“排料阀打开”灯
Drain_closed_disp	Q9.7	BOOL	“排料阀关闭”灯

其它程序元素的符号地址			
符号名	地址	数据类型	描述
EMER_STOP_off	I1.6	BOOL	紧急停车开关
Reset_maint	I1.7	BOOL	所有电机上的维护灯的复位开关
Motor_block	FB1	FB1	用于控制泵和电机的 FB
Valve_block	FC1	FC1	用于控制阀的 FC
DB_feed_pump_A	DB1	FB1	用于控制进料泵 A 的背景数据块
DB_feed_pump_B	DB2	FB1	用于控制进料泵 B 的背景数据块
DB_agitator	DB3	FB1	用于控制搅拌机电机的背景数据块

A.5.2.4 为电机创建 FB

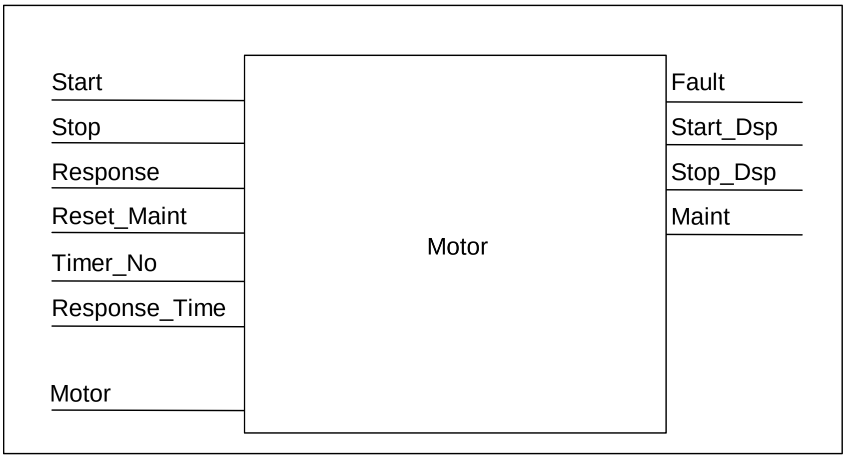
对 FB 有哪些要求?

电机 FB 包含下列逻辑功能:

- 有启动和停止输入。
- 一系列允许设备运行(泵和搅拌机电机)的联锁。联锁的状态("Motor_enable"、"Valve_enable")保存在 OB1 的临时局部数据(L 堆栈)中，当处理电机 FB 时，与启动和停止输出在逻辑上组合在一起。
- 来自设备的反馈必须在一定的时间内显示。否则，会假定发生了错误或故障。然后，该功能会停止电机。
- 必须指定时间点和响应或错误/故障周期的持续时间。
- 如果按下启动按钮且电机已启用，则设备自己开动并运行，直到按下停止按钮。
- 当设备开动时，定时器开始运行。如果在定时器时间用完之前没有收到来自设备的响应信号，则设备停止。

指定输入和输出

下图显示电机通用的 FB 的输入和输出。



定义 FB 的参数

如果使用了用于电机的多重背景 FB(用于控制泵和电机)，就必须为输入和输出定义通用的参数名。

示例过程的电机 FB 要求如下:

- 它必须有来自操作员站的信号，以停止和启动电机和泵。
- 它需要来自电机和泵的响应信号，以指示电机正在运行。

- 它必须计算从发出信号起动电机到收到响应信号之间的时间。如果在该时间中没有收到响应信号，则电机必须关闭。
- 它必须能打开和关闭操作员站的灯。
- 它提供信号以起动电机。

下表所示为示例过程中电机 FB 的参数。下表所示为示例过程中用于电机的 FB 的参数。

参数名称	输入	输出	输入/输出
Start	n		
Stop	n		
Response	n		
Reset_maint	n		
Timer_No	n		
Response_Time	n		
Fault		n	
Start_Dsp		n	
Stop_Dsp		n	
Maint		n	
Motor			n

声明用于电机的 **FB** 的变量

必须声明用于电机的 FB 的输入、输出和输入/输出参数。

地址	声明	名称	类型	初始值
0.0	IN	Start	BOOL	FALSE
0.1	IN	Stop	BOOL	FALSE
0.2	IN	Response	BOOL	FALSE
0.3	IN	Reset_Maint	BOOL	FALSE
2.0	IN	Timer_No	TIMER	
4.0	IN	Response_Time	S5TIME	S5T#0MS
6.0	OUT	Fault	BOOL	FALSE
6.1	OUT	Start_Dsp	BOOL	FALSE
6.2	OUT	Stop_Dsp	BOOL	FALSE
6.3	OUT	Maint	BOOL	FALSE
8.0	IN_OUT	Motor	BOOL	FALSE
10.0	STAT	Time_bin	WORD	W#16#0
12.0	STAT	Time_BCD	WORD	W#16#0
14.0	STAT	Starts	INT	0
16.0	STAT	Start_Edge	BOOL	FALSE

使用 FB 时，输入、输出、输入/输出和静态变量保存在调用语句指定的背景数据块中。临时变量存储在 L 堆栈中。

编程用于电机的 FB

在 STEP 7 中，由其它块调用的每个块都必须先于调用它的块创建。因此，在示例程序中，必须在创建 OB1 之前创建用于电机的 FB。

使用 STL 编程语言的 FB1 代码段如下所示：

Network1 Start/stop and latching

```
A(
O       #Start
O       #Motor
)
AN      #Stop
=       #Motor
```

Network 2 Startup monitoring

```
A       #Motor
L       #Response_Time
SD      #Timer_No
AN      #Motor
R       #Timer_No
L       #Timer_No
T       #Timer_bin
LC      #Timer_No
T       #Timer_BCD
A       #Timer_No
AN      #Response
S       #Fault
R       #Motor
```

Network 3 Start lamp and fault reset

```
A       #Response
=       #Start_Dsp
R       #Fault
```

Network 4 Stop lamp

```
AN      #Response
=       #Stop_Dsp
```

Network 5 Counting the starts

```
A       #Motor
FP      #Start_Edge
JCN     lab1
L       #Starts
+       1
T       #Starts

lab1：NOP      0
```

Network 6 Maintenance lamp

```
L       #Starts
L       50
>=I
=       #Maint
```

Network 7 Reset counter for number of starts

```
A       #Reset_Maint
A       #Maint
JCN     END
L       0
T       #Starts
END:NOP    0
```

创建背景数据块

创建三个数据块，然后逐个打开。在“新建数据块”对话框中，选择“引用功能块的数据块”选项。在“引用”列表框中选择“FB1”。然后数据块被指定为固定分配给 FB1 的背景数据块。

A.5.2.5 创建用于阀的 FC

对 FC 有哪些要求？

用于入口和进料阀以及排料阀的功能包含下列逻辑功能：

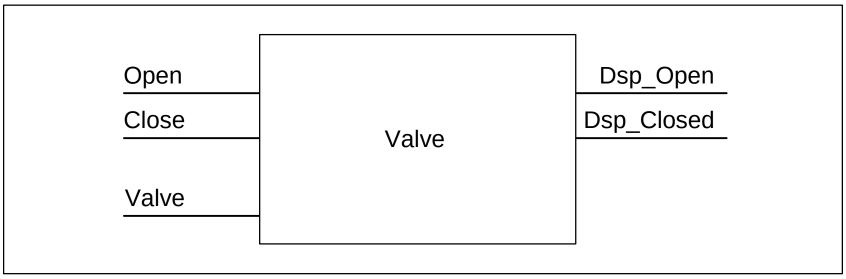
- 有用于开、关阀的输入参数。
- 联锁允许将阀打开。联锁的状态(“Valve_enable”)保存在 OB1 的临时局部数据(L 堆栈)中，在处理阀 FC 时，与开、关输入在逻辑上组合在一起。

下表给出了必须传送给 FC 的参数。

用于阀的参数	输入	输出	输入/输出
Open	✓		
Close	✓		
Dsp_Open		✓	
Dsp_Closed		✓	
Valve			✓

指定输入和输出

下图给出了用于阀的常规 FC 的输入和输出参数。调用用于电机的 FB 的设备传送这些输入参数。阀 FC 返回输出参数。



声明用于阀的 FC 的变量

就象用于电机的 FB 那样，也必须为用于阀的 FC 声明输入、输出和输入/输出参数(参见下面的变量声明表)。

地址	声明	名称	类型	初始值
0.0	IN	Open	BOOL	FALSE
0.1	IN	Close	BOOL	FALSE
2.0	OUT	Dsp_Open	BOOL	FALSE
2.1	OUT	Dsp_Closed	BOOL	FALSE
4.0	IN_OUT	Valve	BOOL	FALSE

对于 FC，临时变量保存在 L 堆栈中。输入、输出和输入/输出变量被保存为指针，指向被调用的 FC 的逻辑块。L 堆栈(在临时变量后)中的附加内存空间用于存储这些变量。

编程用于 FC 的阀

必须在创建 OB1 之前创建用于阀的 FC1 功能，因为被调用块必须在调用块之前创建。

使用 STL 编程语言的 FC1 代码段如下所示：

Network 1 Open/close and latching

```
A(
O      #Open
O      #Valve
)
AN     #Close
=      #Valve
```

Network 2 Display "valve open"

```
A      #Valve
=      #Dsp_Open
```

Network 3 Display "valve closed"

```
AN     #Valve
=      #Dsp_Closed
```

A.5.2.6 创建 OB1

OB1 决定了示例程序的结构。OB1 还包含传送到各种功能的参数，例如：

- 用于进料泵和搅拌器电机的 STL 程序段为用于电机的 FB 提供了用于启动（“Start”）、停止(“Stop”)、响应(“Response”)和复位维护显示（“Reset_Maint”)的输入参数。在 PLC 的每个周期中，都会对用于电机的 FB 进行处理。
- 如果用于电机的 FB 被处理，输入 Timer_No 和 Response_Time 会通知正在使用的定时器的功能，并通知必须测量哪个时间。
- 由于是在 OB1 中被调用，用于阀的 FC 和用于电机的 FB 在可编程控制器的每个程序周期都会被处理。

程序使用具有不同背景数据块的电机 FB 来处理进料泵和搅拌机电机的控制任务。

为 OB1 声明变量

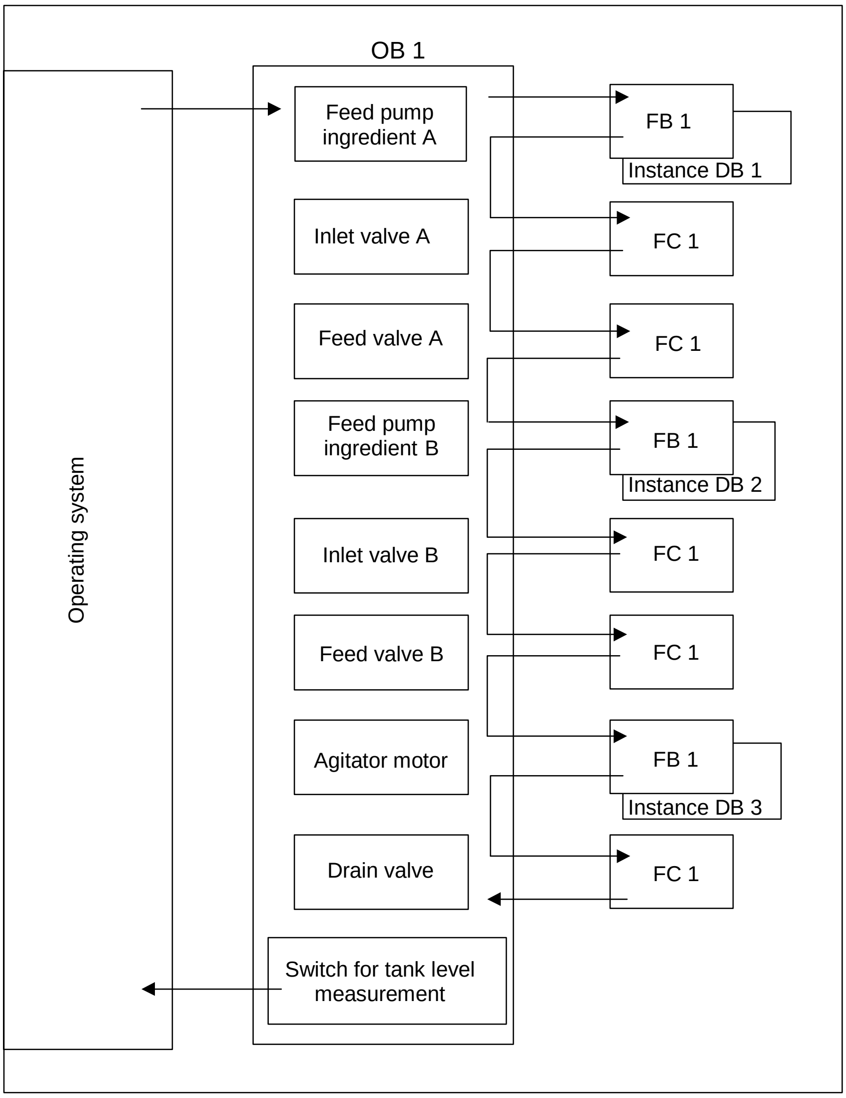
OB1 的变量声明表如下显示。前 20 个字节包含 OB1 的启动信息，不得修改。

地址	声明	名称	类型
0.0	TEMP	OB1_EV_CLASS	BYTE
1.0	TEMP	OB1_SCAN1	BYTE
2.0	TEMP	OB1_PRIORITY	BYTE
3.0	TEMP	OB1_OB_NUMBR	BYTE
4.0	TEMP	OB1_RESERVED_1	BYTE
5.0	TEMP	OB1_RESERVED_2	BYTE
6.0	TEMP	OB1_PREV_CYCLE	INT
8.0	TEMP	OB1_MIN_CYCLE	INT
10.0	TEMP	OB1_MAX_CYCLE	INT
12.0	TEMP	OB1_DATE_TIME	DATE_AND_TIME
20.0	TEMP	Enable_motor	BOOL
20.1	TEMP	Enable_valve	BOOL
20.2	TEMP	Start_fulfilled	BOOL
20.3	TEMP	Stop_fulfilled	BOOL
20.4	TEMP	Inlet_valve_A_open	BOOL
20.5	TEMP	Inlet_valve_A_closed	BOOL
20.6	TEMP	Feed_valve_A_open	BOOL
20.7	TEMP	Feed_valve_A_closed	BOOL
21.0	TEMP	Inlet_valve_B_open	BOOL
21.1	TEMP	Inlet_valve_B_closed	BOOL
21.2	TEMP	Feed_valve_B_open	BOOL
21.3	TEMP	Feed_valve_B_closed	BOOL
21.4	TEMP	Open_drain	BOOL
21.5	TEMP	Close_drain	BOOL
21.6	TEMP	Valve_closed_fulfilled	BOOL

为 OB1 创建程序

在 STEP 7 中，由其它块调用的每个块都必须先于调用它的块创建。因此，在示例程序中，必须在 OB1 编程前，创建用于电机的 FB 和用于阀的 FC。

在 OB1 中，多次调用块 FB1 和 FC1；调用 FB1 使用了不同的背景数据块：



在 STL 编程语言中，OB1 的代码段如下所示：

Network 1 Interlocks for feed pump A

```
A       "EMER_STOP_off"
A       "Tank_below_max"
AN      "Drain"
=       #Enable_Motor
```

Network 2 Calling FB Motor for ingredient A

```
A               "Feed_pump_A_start"
A               #Enable_Motor
=               #Start_Fulfilled
A(
O               "Feed_pump_A_stop"
ON      #Enable_Motor
)
=               #Stop_Fulfilled
CALL    "Motor_block", "DB_feed_pump_A"
        Start     :=#Start_Fulfilled
        Stop      :=#Stop_Fullfilled
        Response        :="Flow_A"
        Reset_Maint     :="Reset_maint"
        Timer_No        :=T12
        Reponse_Time:=S5T#7S
        Fault     :="Feed_pump_A_fault"
        Start_Dsp       :="Feed_pump_A_on"
        Stop_Dsp        :="Feed_pump_A_off"
        Maint     :="Feed_pump_A_maint"
        Motor     :="Feed_pump_A"
```

Network 3 Delaying the valve enable ingredient A

```
A       "Feed_pump_A"
L       S5T#1S
SD      T       13
AN      "Feed_pump_A"
R       T       13
A       T       13
=       #Enable_Valve
```

Network 4 Inlet valve control for ingredient A

```
AN      "Flow_A"
AN      "Feed_pump_A"
=               #Close_Valve_Fulfilled
CALL    "Valve_block"
        Open      :=#Enable_Valve
        Close     :=#Close_Valve_Fulfilled
        Dsp_Open        :=#Inlet_Valve_A_Open
        Dsp_Closed:=#Inlet_Valve_A_Closed
        Valve     :="Inlet_Valve_A"
```

Network 5 Feed valve control for ingredient A

```
AN      "Flow_A"
AN      "Feed_pump_A"
=               #Close_Valve_Fulfilled
CALL    "Valve_block"
        Open      :=#Enable_Valve
        Close     :=#Close_Valve_Fulfilled
        Dsp_Open        :=#Feed_Valve_A_Open
        Dsp_Closed:=#Feed_Valve_A_Closed
        Valve     :="Feed_Valve_A"
```

Network 6 Interlocks for feed pump B

```
A       "EMER_STOP_off"
A       "Tank_below_max"
AN      "Drain"
=       "Enable_Motor
```

Network 7 Calling FB Motor for ingredient B

```
A               "Feed_pump_B_start"
A               #Enable_Motor
=               #Start_Fulfilled
A(
O               "Feed_pump_B_stop"
ON      #Enable_Motor
)
=               #Stop_Fulfilled
CALL    "Motor_block", "DB_feed_pump_B"
        Start     :=#Start_Fulfilled
        Stop      :=#Stop_Fullfilled
        Response        :="Flow_B"
        Reset_Maint     :="Reset_maint"
        Timer_No        :=T14
        Reponse_Time:=S5T#7S
        Fault     :="Feed_pump_B_fault"
        Start_Dsp       :="Feed_pump_B_on"
        Stop_Dsp        :="Feed_pump_B_off"
        Maint     :="Feed_pump_B_maint"
        Motor     :="Feed_pump_B"
```

Network 8 Delaying the valve enable ingredient B

```
A       "Feed_pump_B"
L       S5T#1S
SD      T       15
AN      "Feed_pump_B"
R       T       15
A       T       15
=       #Enable_Valve
```

Network 9 Inlet valve control for ingredient B

```
AN      "Flow_B"
AN      "Feed_pump_B"
=               #Close_Valve_Fulfilled
CALL    "Valve_block"
        Open      :=#Enable_Valve
        Close     :=#Close_Valve_Fulfilled
        Dsp_Open        :=#Inlet_Valve_B_Open
        Dsp_Closed:=#Inlet_Valve_B_Closed
        Valve     :="Inlet_Valve_B"
```

Network 10 Feed valve control for ingredient B

```
AN      "Flow_B"
AN      "Feed_pump_B"
=               #Close_Valve_Fulfilled
CALL    "Valve_block"
        Open    :=#Enable_Valve
        Close   :=#Close_Valve_Fulfilled
        Dsp_Open        :=#Feed_Valve_B_Open
        Dsp_Closed:=#Feed_Valve_B_Closed
        Valve   :="Feed_Valve_B"
```

Network 11 Interlocks for agitator

```
A       "EMER_STOP_off"
A       "Tank_above_min"
AN      "Drain"
=       #Enable_Motor
```

Network 12 Calling FB Motor for agitator

```
A               "Agitator_start"
A               #Enable_Motor
=               #Start_Fulfilled
A(
O               "Agitator_stop"
ON      #Enable_Motor
)
=               #Stop_Fulfilled
CALL    "Motor_block", "DB_Agitator"
        Start   :=#Start_Fulfilled
        Stop    :=#Stop_Fullfilled
        Response        :="Agitator_running"
        Reset_Maint     :="Reset_maint"
        Timer_No        :=T16
        Reponse_Time:=S5T#10S
        Fault   :="Agitator_fault"
        Start_Dsp       :="Agitator_on"
        Stop_Dsp        :="Agitator_off"
        Maint   :="Agitator_maint"
        Motor   :="Agitator"
```

Network 13 Interlocks for drain valve

```
A       "EMER_STOP_off"
A       "Tank_not_empty"
AN      "Agitator"
=       "Enable_Valve
```

Network 14　　Drain valve control

```
A       "Drain_open"
A       #Enable_Valve
=       #Open_Drain
A(
O       "Drain_closed"
ON      #Enable_Valve
)
=       #Close_Drain
CALL    "Valve_block"
        Open        :=#Open_Drain
        Close       :=#Close_Drain
        Dsp_Open    :="Drain_open_disp"
        Dsp_Closed  :="Drain_closed_disp"
        Valve       :="Drain"
```

Network 15　　Tank level display

```
AN      "Tank_below_max"
=       "Tank_max_disp"
AN      "Tank_above_min"
=       "Tank_min_disp"
AN      "Tank_not_empty"
=       "Tank_empty_disp"
```

A.5.3 处理时间中断的实例

A.5.3.1 处理时间中断的实例

用户程序“时间中断”的结构

FC12

OB10

OB1 和 OB80

A.5.3.2 用户程序“时间中断”的结构

任务

从周一上午 5 点到周五晚上 8 点的时间段中，输出点 Q 4.0 应该置位。从周五晚上 8 点到周一上午 5 点的时间段中，输出点 Q 4.0 应该复位。

转换成用户程序

下表说明所使用的块的子任务。

块	子任务
OB1	调用功能 FC12
FC12	根据输出 Q 4.0 的状态、时间中断状态和输入 I 0.0 与 I 0.1 指定启动时间 设置时间中断 激活时间中断 CAN_TINT
OB10	根据当前周时 指定启动时间 置位或复位输出 Q 4.0 设置下一个时间中断 激活下一个时间中断
OB80	置位输出 Q 4.1 将 OB80 的启动事件信息存储在位存储区域

使用的地址

下表给出了所使用的共享地址。临时局部变量在各个块的声明部分进行声明。

地址	含义
I0.0	输入，可启用“设置时间中断”和“激活时间中断”
I0.1	输入，取消时间中断
Q4.0	输出，通过时间中断 OB(OB10)实现置位/复位
Q4.1	输出，由时间错误(OB80)置位
MW16	时间中断(SFC31“QRY_TINT”)的 STATUS
MB100 到 MB107	用于 OB10 的启动事件信息的存储器(仅适用于时间)
MB110 到 MB129	用于 OB80 的启动事件信息的存储器(时间错误)
MW200	SFC28“SET_TINT”的 RET_VAL
MB202	SFC 的二进制结果(状态位 BR)缓存区
MW204	SFC30“ACT_TINT”的 RET_VAL
MW208	SFC31“QRY_TINT”的 RET_VAL

系统功能和所使用的功能

编程实例中用到了下列系统功能：

- SFC28“SET_TINT” ： 设置时间中断
- SFC29“CAN_TINT” ： 取消时间中断
- SFC30“ACT_TINT” ： 激活时间中断
- SFC31“QRY_TINT” ： 查询时间中断
- FC3“D_TOD_DT” ： 将 DATE 和 TIME_OF_DAY 组合为 DT

A.5.3.3 FC12

声明部分

下列临时局部变量在 FC12 的声明部分进行声明：

变量名称	数据类型	声明	注释
IN_TIME	TIME_OF_DAY	TEMP	启动时间
IN_DATE	DATE	TEMP	启动日期
OUT_TIME_DATE	DATE_AND_TIME	TEMP	启动转换后的日期/时间
OK_MEMORY	BOOL	TEMP	允许设置时间中断

STL 代码段

在 FC12 的代码段中输入下列 STL 用户程序：

```
STL(FC12)                                           解释
Network 1
       CALL SFC 31                                  SFC QRY_TINT
        OB_NO  := 10                                查询时间中断的状态
        RET_VAL:= MW 208
        STATUS := MW 16

Network 2:
       AN      Q       4.0                          根据 Q 4.0 指定启动时间
       JC      mond                                 (在变量#IN_DATE 和#IN_TIME 中)
       L       D#1995-1-27                          启动日期是周五
       T       #IN_DATE
       L       TOD#20:0:0.0
       T       #IN_TIME
       JU      cnvrt
mond:  L       D#1995-1-23                          启动日期是周一
       T       #IN_DATE
       L       TOD#5:0:0.0
       T       #IN_TIME
cnvrt: NOP     0
```

STL(FC12)	解释
Network 3:	
CALL FC 3	将格式从 DATE 和 TIME_OF_DAY 转换为 DATE_AND_TIME(用于设置时间中断)
IN1 := #IN_DATE	
IN2 := #IN_TIME	
RET_VAL := #OUT_TIME_DATE	
Network 4:	设置时间中断的所有要求是否已满足？(输入用于启用设置的输入、时间中断没有激活并且时间中断 OB 已载入)
A I 0.0	
AN M 17.2	
A M 17.4	如果是这样，设置时间中断...
= #OK_MEMORY	
Network 5:	
A #OK_MEMORY	
JNB m001	
CALL SFC 28	
OB_NO := 10	
SDT := #OUT_TIME_DATE	
PERIOD := W#16#1201	
RET_VAL := MW 200	...并激活时间中断。
m001 A BR	
= M 202.3	
Network 6:	
A #OK_MEMORY	
JNB m002	
CALL SFC 30	
OB_NO := 10	
RET_VAL := MW 204	如果用于取消时间中断的输入置位，则取消时间中断。
m002 A BR	
= M 202.4	
Network 7:	
A I 0.1	
JNB m003	
CALL SFC 29	
OB_NO := 10	
RET_VAL := MW 210	
m003 A BR	
= M 202.5	

A.5.3.4 OB10

声明部分

和 OB10 的缺省声明部分不同，声明了下列临时局部变量：

- 用于全部启动事件信息的结构(STARTINFO)
- 在 STARTINFO 结构内，用于存储时间的结构(T_STMP)
- 其它临时局部变量 WDAY、IN_DATE、IN_TIME 和 OUT_TIME_DATE

变量名称	数据类型	声明	注释
STARTINFO	STRUCT	TEMP	OB10 的全部启动事件信息声明为结构
E_ID	WORD	TEMP	事件标识号：
PR_CLASS	BYTE	TEMP	优先级等级
OB_NO	BYTE	TEMP	OB 编号
RESERVED_1	BYTE	TEMP	保留
RESERVED_2	BYTE	TEMP	保留

变量名称	数据类型	声明	注释
PERIOD	WORD	TEMP	时间中断的周期
RESERVED_3	DWORD	TEMP	保留
T_STMP	STRUCT	TEMP	存储详细时间的结构
YEAR	BYTE	TEMP	
MONTH	BYTE	TEMP	
DAY	BYTE	TEMP	
HOUR	BYTE	TEMP	
MINUTES	BYTE	TEMP	
SECONDS	BYTE	TEMP	
MSEC_WDAY	WORD	TEMP	
	END_STRUCT	TEMP	
	END_STRUCT	TEMP	
WDAY	INT	TEMP	周时
IN_DATE	DATE	TEMP	FC3 的输入变量 (时间格式转换)
IN_TIME	TIME_OF_DAY	TEMP	FC3 的输入变量 (时间格式转换)
OUT_TIME_DATE	DATE_AND_TIME	TEMP	FC3 的输出变量和 SFC28 的输入变量

STL 代码段

在 OB10 的代码段中输入下列 STL 用户程序：

STL(OB10)	解释
Network 1	
L #STARTINFO.T_STMP.MSEC_WDAY	选择周时
L W#16#F	
AW	
T #WDAY	并存储。
Network 2:	
L #WDAY	如果不是周一，则指定周一上午 5 点为下一个启动时间，并复位输出 Q 4.0。
L 2	
<>I	
JC mond	
Network 3:	否则，如果是周一，则指定周五晚上 8:00(20:00)为下一个启动时间，并置位输出 Q 4.0。
L D#1995-1-27	
T #IN_DATE	
L TOD#20:0:0.0	
T #IN_TIME	
SET	
= Q 4.0	
JU cnvrt	
mond: L D#1995-1-23	
T #IN_DATE	
L TOD#5:0:0.0	
T #IN_TIME	指定启动时间。
CLR	将指定的启动时间转换为 DATE_AND_TIME 格式(对于 SFC28)。
= Q 4.0	
cnvrt: NOP 0	
Network 4:	设置时间中断。
CALL FC 3	
IN1 := #IN_DATE	
IN2 := #IN_TIME	
RET_VAL := #OUT_TIME_DATE	
Network 5:	
CALL SFC 28	
OB_NO := 10	
SDT := #OUT_TIME_DATE	
PERIOD := W#16#1201	
RET_VAL := MW 200	
A BR	
= M 202.1	
Network 6:	
CALL SFC 30	激活时间中断
OB_NO := 10	
RET_VAL := MW 204	
A BR	
= M 202.2	
Network 7:	
CALL SFC 20	块传送：将 OB10 的启动事件信息中的时间保存到存储器区 MB100 至 MB107 中。
SRCBLK := #STARTINFO.T_STMP	
RET_VAL := MW 206	
DSTBLK := P#M 100.0 BYTE 8	

A.5.3.5 OB1 和 OB80

因为 OB1 (用于循环程序的 OB)的启动事件信息在此实例中未作评价，所以只显示 OB80 的启动事件信息。

OB1 代码段

在 OB1 的代码段中输入下列 STL 用户程序：

STL(OB1)	解释
CALL FC 12	调用功能 FC12

OB80 声明部分

和 OB80 的缺省声明部分不同，声明了下列临时局部变量：

- 用于全部启动事件信息的结构(STARTINFO)
- 在 STARTINFO 结构内，用于存储时间的结构(T_STMP)

变量名称	数据类型	声明	注释
STARTINFO	STRUCT	TEMP	OB80 的全部启动事件信息声明为结构
E_ID	WORD	TEMP	事件标识号：
PR_CLASS	BYTE	TEMP	优先级等级
OB_NO	BYTE	TEMP	OB 编号
RESERVED_1	BYTE	TEMP	保留
RESERVED_2	BYTE	TEMP	保留
A1_INFO	WORD	TEMP	引起错误的事件的有关附加信息
A2_INFO	DWORD	TEMP	错误的事件标识号、优先级和 OB 编号的有关附加信息。
T_STMP	STRUCT	TEMP	存储详细时间的结构
YEAR	BYTE	TEMP	
MONTH	BYTE	TEMP	
DAY	BYTE	TEMP	
HOUR	BYTE	TEMP	
MINUTES	BYTE	TEMP	
SECONDS	BYTE	TEMP	
MSEC_WDAY	WORD	TEMP	
	END_STRUCT	TEMP	
	END_STRUCT	TEMP	

OB80 代码段

在如果发生时间错误，则操作系统将调用的 OB80 的代码段中输入下列 STL 用户程序。

STL(OB80)	解释
Network 1	
AN Q 4.1	如果发生时间错误，置位输出 Q 4.1。
S Q 4.1	
CALL SFC 20 SRCBLK := #STARTINFO RET_VAL := MW 210 DSTBLK := P#M 110.0 Byte 20	块传送：将全部启动事件信息保存到存储器区 MB110 至 MB129。

A.5.4 处理时间中断的实例

A.5.4.1 处理时间中断的实例

用户程序“延时中断”的结构

OB20

OB1

A.5.4.2 用户程序“延时中断”的结构

任务

当置位输入 I 0.0 时，应该在 10 秒以后置位输出 Q 4.0。每次置位输入 I 0.0 后，延迟时间应该重新开始。

延时中断的启动时间(秒和毫秒)应该在延时中断 OB(OB20)的启动事件信息中作为用户特定的标识号出现。

如果 I 0.1 在这 10 秒内置位，则组织块 OB20 不应该被调用；这意味着输出 Q 4.0 不应被置位。

当输入 I 0.2 置位时，输出 Q 4.0 应复位。

转换成用户程序

下表说明了所使用的块的子任务。

块	子任务
OB1	读取当前时间并准备启动延时中断 根据输入 I 0.0 上的边沿变化，启动延时中断 根据延时中断的状态和输入 I 0.1 上的边沿变化，取消延时中断 根据输入 I 0.2 的状态，复位输出 Q 4.0
OB20	置位输出 Q 4.0 读取和准备当前时间 将启动事件信息保存到位存储区域

使用的地址

下表显示所使用的共享地址。临时局部变量在各个块的声明部分进行声明。

地址	含义
I0.0	输入，用于启用“启动延时中断”
I0.1	输入，用于取消延时中断
I0.2	输入，用于复位输出 Q 4.0
Q4.0	输出，由延时中断 OB(OB20)置位
MB1	用作 SFC 的边沿标志和二进制结果(状态位 BR)缓存区
MW4	延时中断(SFC34“QRY_TINT”)的 STATUS
MD10	来自 OB1 的启动事件信息的 BCD 编码的秒和毫秒
MW 100	SFC32“SRT_DINT”的 RET_VAL
MW102	SFC34“QRY_DINT”的 RET_VAL
MW104	SFC33“CAN_DINT”的 RET_VAL
MW106	SFC20“BLKMOV”的 RET_VAL
MB120 至 MB139	OB20 启动事件信息的存储区
MD140	来自 OB20 的启动事件信息的 BCD 编码的秒和毫秒
MW144	来自 OB1 启动事件信息的 BCD 编码的秒和毫秒；从 OB20 (用户特定的标识符 SIGN)的启动事件信息中获得

所使用的系统功能

在用户程序“延时中断”中使用下列 SFC：

- SFC32“SRT_DINT” ： 启动延时中断
- SFC33“CAN_DINT” ： 取消延时中断
- SFC34“QRY_DINT” ： 查询延时中断的状态

A.5.4.3 OB20

声明部分

与 OB20 的缺省声明部分不同，将对下列临时局部变量进行声明：

- 用于整个启动事件信息的结构(STARTINFO)
- 在 STARTINFO 结构内的时间结构(T_STMP)

变量名称	数据类型	声明	注释
STARTINFO	STRUCT	TEMP	OB20 的启动信息
E_ID	WORD	TEMP	事件标识号：
PC_NO	BYTE	TEMP	优先级等级
OB_NO	BYTE	TEMP	OB 编号
D_ID1	BYTE	TEMP	数据标识号 1
D_ID 2	BYTE	TEMP	数据标识号 2
SIGN	WORD	TEMP	用户特定的标识号
DTIME	TIME	TEMP	延时中断的启动时间
T_STMP	STRUCT	TEMP	存储详细时间的结构 (时间标志)
YEAR	BYTE	TEMP	
MONTH	BYTE	TEMP	
DAY	BYTE	TEMP	
HOUR	BYTE	TEMP	
MINUTES	BYTE	TEMP	
SECONDS	BYTE	TEMP	
MSEC_WDAY	WORD	TEMP	
	END_STRUCT	TEMP	
	END_STRUCT	TEMP	

代码段

在 OB20 的代码段中输入下列 STL 用户程序：

STL(OB20)	解释
Network 1	
SET	无条件地置位输出 Q 4.0
= Q 4.0	
Network 2:	
L QW 4	立即激活输出字
T PQW 4	
Network 3:	
L #STARTINFO.T_STMP.SECONDS	从启动事件信息中读取秒
T MW 140	从启动事件信息中读取毫秒和周时
L #STARTINFO.T_STMP.MSEC_WDAY	
T MW 142	清除周时并
L MD 140	写回成毫秒(现在 MW 142 中为 BCD 编码的数据)
SRD 4	从启动事件信息中读取延时中断的启动时间(=调用 SFC32)
T MD 140	
Network 4:	将启动事件信息复制到存储器区(MB120 至 MB139)
L #STARTINFO.SIGN	
T MW 144	
Network 5:	
CALL SFC 20	
SRCBLK := STARTINFO	
RET_VAL := MW 106	
DSTBLK := P#M 120.0 Byte 20	

A.5.4.4 OB1

声明部分

和 OB1 的缺省声明不同，声明下列临时局部变量：

- 用于全部启动事件信息的结构(STARTINFO)
- 在 STARTINFO 结构内，用于存储时间的结构(T_STMP)

变量名称	数据类型	声明	注释
STARTINFO	STRUCT	TEMP	为 OB1 启动信息
E_ID	WORD	TEMP	事件标识号：
PC_NO	BYTE	TEMP	优先级等级
OB_NO	BYTE	TEMP	OB 编号
D_ID1	BYTE	TEMP	数据标识号 1
D_ID 2	BYTE	TEMP	数据标识号 2
CUR_CYC	INT	TEMP	当前周期
MIN_CYC	INT	TEMP	最小周期
MAX_CYC	INT	TEMP	最大周期
T_STMP	STRUCT	TEMP	存储详细时间的结构(时间标记)
YEAR	BYTE	TEMP	
MONTH	BYTE	TEMP	

变量名称	数据类型	声明	注释
DAY	BYTE	TEMP	
HOUR	BYTE	TEMP	
MINUTES	BYTE	TEMP	
SECONDS	BYTE	TEMP	
MSEC_WDAY	WORD	TEMP	
	END_STRUCT	TEMP	
	END_STRUCT	TEMP	

代码段

在 OB1 的代码段中输入下列 STL 用户程序：

```
STL(OB1)                                              解释
Network 1
       L      #STARTINFO.T_STMP.SECONDS               从启动事件信息中读取秒
       T      MW 10                                   从启动事件信息中读取毫秒和周时
       L      #STARTINFO.T_STMP.MSEC_WDAY             清除周时并
       T      MW 12                                   写回成毫秒(现在 MW 12 中为 BCD 编码的数据)
       L      MD 10                                   是输入 I 0.0 的上升沿吗?
       SRD    4
       T      MD 10
Network 2:
       A      I 0.0                                   如果是，则启动延时中断(延时中断的启动时间分配给
       FP     M 1.0                                   参数 SIGN)
       =      M 1.1
Network 3:
       A      M 1.1
       JNB    m001
       CALL SFC 32
        OB_NO  := 20
        DTME   := T#10S                               查询延时中断(SFC QRY_DINT)的状态
        SIGN   := MW 12
        RET_VAL:= MW 100
m001:  NOP    0
                                                      是输入 I 0.1 的上升沿吗?
Network 4:
       CALL SFC 34
        OB_NO  := 20
        RET_VAL:= MW 102
        STATUS := MW 4                                ...并激活延时中断(延时中断 STATUS 的第 2 位)?
Network 5:                                            然后取消延时中断
       A      I 0.1
       FP     M 1.3
       =      M 1.4
                                                      用输入 I 0.2 复位输出 Q 4.0
Network 6:
       A      M 1.4
       A      M 5.2
       JNB    m002
       CALL SFC 33
        OB_NO  := 20
        RET_VAL:= MW 104
m002:  NOP    0
       A      I 0.2
       R      Q 4.0
```

A.5.4.5 屏蔽和取消屏蔽同步错误的实例

下面的用户程序实例说明了如何屏蔽和取消屏蔽同步错误。使用 SFC36 “MSK_FLT”，下列错误在编程错误过滤器中被屏蔽：

- 读取数据时发生的区域长度错误
- 写入数据时发生的区域长度错误

第二次调用 SFC36“MSK_FLT”时，访问区域也可以被屏蔽：

- 写入数据时发生的 I/O 访问错误

使用 SFC38“READ_ERR”，查询被屏蔽的同步错误。用 SFC37“DMSK_FLT”再次取消屏蔽“写入数据时发生的 I/O 访问错误”。

代码段

在下面将看到的 OB1 中，用户程序实例是在语句表中编程的。

STL(Network 1)	解释
AN M 255.0	非掉电保护的存储位 M 255.0 (仅在第一次运行=0)
JNB m001	SFC36 MSK_FLT (屏蔽同步错误)
CALL SFC 36	位 2 = 位 3 = 1 (BLFL 和 BLFS 被屏蔽)
	所有位=0 (没有访问错误被屏蔽)
PRGFLT_SET_MASK :=DW#16#C	返回值
	输出当前编程错误过滤器到 MD10
ACCFLT_SET_MASK :=DW#16#0	输出当前访问错误过滤器到 MD14
RET_VAL :=MW 100	如果屏蔽成功则置位 M255.0
PRGFLT_MASKED :=MD 10	
ACCFLT_MASKED :=MD 14	
m001: A BR	
S M 255.0	

STL(Network 2)	解释
CALL SFC 36	SFC36 MSK_FLT (屏蔽同步错误)
	所有位=0 (不再有编程错误被屏蔽)
PRGFLT_SET_MASK :=DW#16#0	位 3=1 (写访问错误被屏蔽)
	返回值
ACCFLT_SET_MASK :=DW#16#8	输出当前编程错误过滤器到 MD20
	输出当前访问错误过滤器到 MD24
RET_VAL :=MW 102	
PRGFLT_MASKED :=MD 20	
ACCFLT_MASKED :=MD 24	

STL(Network 3)	解释
AN M 27.3	如果写访问错误(ACCFLT_MASKED 中的位 3)没有屏
BEC	蔽，块结束

STL(Network 4)	解释
L B#16#0	
T PQB 16	到 PQB 16 的写访问(通过值 0)

STL(Network 5)			解释
CALL	SFC 38		SFC38 READ_ERR(查询同步错误)
			所有位=0 (没有查询到编程错误)
	PRGFLT_QUERY	:=DW#16#0	位 3=1 (查询到写访问错误)
			返回值
	ACCFLT_QUERY	:=DW#16#8	输出当前编程错误过滤器到 MD30
			输出当前访问错误过滤器到 MD34
	RET_VAL	:=MW 104	没有发生错误，且没有检测到写访问错误
	PRGFLT_CLR	:=MD 30	
			取反 RLO
	ACCFLT_CLR	:=MD 34	如果 PQB 16 存在，M 0.0=1
A	BR		
A	M 37.3		
NOT			
=	M 0.0		

STL(Network 6)		解释
L	B#16#0	
T	PQB 17	到 PQB 17 的写访问(通过值 0)

STL(Network 7)			解释
CALL	SFC 38		SFC38 READ_ERR (查询同步错误)
			所有位=0(没有查询到编程错误)
	PRGFLT_QUERY	:=DW#16#0	位 3=1 (查询到写访问错误)
			返回值
	ACCFLT_QUERY	:=DW#16#8	输出当前编程错误过滤器到 MD30
			输出当前访问错误过滤器到 MD34
	RET_VAL	:=MW 104	没有发生错误，且没有检测到写访问错误
	PRGFLT_CLR	:=MD 30	
			取反 RLO
	ACCFLT_CLR	:=MD 34	如果 PQB 17 存在，M 0.1=1
A	BR		
A	M 37.3		
NOT			
=	M 0.1		

STL(Network 8)		解释
L	B#16#0	
T	PQB 18	到 PQB 18 的写访问(通过值 0)

STL(Network 9)			解释
CALL	SFC 38		SFC38 READ_ERR (查询同步错误)
			所有位=0 (没有查询到编程错误)
	PRGFLT_QUERY	:=DW#16#0	位 3=1 (查询到写访问错误)
			返回值
	ACCFLT_QUERY	:=DW#16#8	输出当前编程错误过滤器到 MD30
			输出当前访问错误过滤器到 MD34
	RET_VAL	:=MW 104	没有发生错误，且没有检测到写访问错误
	PRGFLT_CLR	:=MD 30	
			取反 RLO
	ACCFLT_CLR	:=MD 34	如果 PQB 18 存在，M 0.2=1
A	BR		
A	M 37.3		
NOT			
=	M 0.2		

STL(Network 10)		解释
L	B#16#0	
T	PQB 19	到 PQB 19 的写访问(通过值 0)

STL(Network 11)	解释
CALL SFC 38	SFC38 READ_ERR (查询同步错误)
	所有位=0 (没有查询到编程错误)
PRGFLT_QUERY :=DW#16#0	位 3=1 (查询到写访问错误)
	返回值
ACCFLT_QUERY :=DW#16#8	输出当前编程错误过滤器到 MD30
	输出当前访问错误过滤器到 MD34
RET_VAL :=MW 104	没有发生错误，且没有检测到写访问错误
PRGFLT_CLR :=MD 30	
	取反 RLO
ACCFLT_CLR :=MD 34	如果 PQB 19 存在，M 0.3=1
A BR	
A M 37.3	
NOT	
= M 0.3	

STL(Network 12)	解释
CALL SFC 37	SFC37 DMSK_FLT (取消屏蔽同步错误)
	所有位=0 (不再有编程错误被取消屏蔽)
PRGFLT_RESET_MASK :=DW#16#0	位 3=1 (写访问错误被取消屏蔽)
	返回值
ACCFLT_RESET_MASK :=DW#16#8	输出当前编程错误过滤器到 MD20
	输出当前访问错误过滤器到 MD24
RET_VAL :=MW 102	
PRGFLT_MASKED :=MD 20	
ACCFLT_MASKED :=MD 24	

STL(Network 13)	解释
A M 27.3	如果写访问错误(ACCFLT_MASKED 中的位 3)没有取
BEC	消屏蔽，块结束

STL(Network 14)	解释
A M 0.0	
JNB m002	
L IB 0	如果存在，传送 IB0 到 PQB 16
T PQB 16	
m002: NOP 0	

STL(Network 15)	解释
A M 0.1	
JNB m003	
L IB 1	如果存在，传送 IB1 到 PQB17
T PQB 17	
m003: NOP 0	

STL(Network 16)	解释
A M 0.2	
JNB m004	
L IB 2	如果存在，传送 IB2 到 PQB18
T PQB 18	
m004: NOP 0	

STL(Network 17)	解释
A M 0.3	
JNB m005	
L IB 3	如果存在，传送 IB3 到 PQB19
T PQB 19	
m005: NOP 0	

A.5.4.6 禁止和允许中断和异步错误的实例(SFC39 和 SFC40)

在此用户程序实例中，假定程序段不能由中断程序中断。对于此程序段，使用 SFC 39“DIS_IRT”禁止 OB35 调用(时间中断)，然后使用 SFC 40“EN_IRT”再次启用 OB35 调用。

OB1 中调用了 SFC39 和 SFC40：

STL(OB1)		解释
A	M 0.0	可以被中断，而不会产生问题的程序段：
S	M 90.1	
A	M 0.1	
S	M 90.0	
:		不得由中断程序中断的程序段：
:		禁止并放弃中断
		模式 2：禁止单个中断 OB
CALL	SFC 39	禁止 OB35
MODE	:=B#16#2	
OB_NO	:=35	
RET_VAL	:=MW 100	
:		
:		
L	PIW 100	
T	MW 200	
L	MW 90	
T	MW 92	
:		允许中断
:		模式 2：允许单个中断 OB
CALL	SFC 40	允许 OB35
MODE	:=B#16#2	
OB_NO	:=35	可以被中断，而不会产生问题的程序段：
RET_VAL	:=MW 102	
A	M 10.0	
S	M 190.1	
A	M 10.1	
S	M 190.0	
:		
:		

A.5.4.7 对中断和异步错误进行延迟处理的实例(SFC41 和 SFC40)

在此用户程序实例中，假定程序段不能由中断程序中断。对于此程序段，使用 SFC41“DIS_AIRT”延迟中断，随后再用 SFC42“EN_AIRT”启用。

在 OB1 中调用了 SFC41 和 SFC42：

STL(OB1)			解释
	A	M 0.0	可以被中断，而不会产生问题的程序段：
	S	M 90.1	
	A	M 0.1	
	S	M 90.0	
	:		不得由中断程序中断的程序段：
	:		禁用和延迟中断
	CALL	SFC 41	
	RET_VAL	:=MW 100	
	L	PIW 100	
	T	MW 200	
	L	MW 90	
	T	MW 92	
	:		
	:		允许中断
	:		
	CALL	SFC 42	所设置的中断禁用的数目在返回值中。
	RET_VAL	:=MW 102	
	L	MW 100	所设置的中断禁用的数目在返回值中。 在中断启用后，此数目必须具有
	DEC	1	同中断禁用前(此处为“0”)相同的值
	L	MW 102	
	<>I		可以被中断，而不会产生问题的程序段：
	JC	err	
	A	M 10.0	
	S	M 190.1	显示所设置的中断禁用的数目
	A	M 10.1	
	S	M 190.0	
	:		
	:		
	BEU		
err:	L	MW 102	
	T	QW 12	

A.6 存取进程和 I/O 数据区

A.6.1 访问过程数据区

CPU 既可间接使用过程映像表，也可直接通过底板/P 总线来访问集中和分布式数字输入/输出模块的输入和输出。

CPU 可通过底板/P 总线直接访问集中和分布式模拟输入/输出模块的输入和输出。

对模块进行寻址

当使用 STEP 7 对模块进行组态时，可按以下规则将程序中使用的地址分配给模块：

- 对于集中 I/O 模块：在组态表中进行机架的分布以及模块到插槽的分配
- 对于具有分布式 I/O (PROFIBUS DP 或 PROFINET IO) 的工作站：在“主站系统”组态表中按 PROFIBUS 地址排列 DP 从站，并将模块分配给插槽。

通过对模块进行组态，将不必再使用开关来设定每个模块上的地址。组态完成后，编程设备将把数据发送给 CPU，从而使该 CPU 能够识别为其分配的模块。

外设 I/O 寻址

对于输入和输出，存在一个单独的地址区。这意味着外设区的地址不仅必须包含有字节或字访问类型，而且必须包含有用于输入的 I 标识符和用于输出的 Q 标识符。

下表列出了可供使用的外设地址区。

地址区	通过下列大小的单元进行访问	S7 符号(IEC)
外设(I/O)区：输入	外设输入字节 外设输入单字 外设输入双字	PIB PIW PID
外设(I/O)区：输出	外设输出字节 外设输入双字 外设输出双字	PQB PQW PQD

为找出各个模块上有哪个地址区可用，请参见下列手册：

- “S7-300 可编程控制器，硬件与安装”手册
- “S7-300、M7-300 可编程控制器，模块技术规格说明”参考手册
- S7-400、M7-400 可编程控制器，模块技术规格说明”参考手册

模块起始地址

模块起始地址就是模块的最低字节地址。它表示模块用户数据区的起始地址，且在很多情况下被用来表示整个模块。

例如，模块起始地址将输入到相应组织块启动信息中的硬件中断、诊断中断、插入/拆卸模块中断以及电源故障中断，并将用于标识启动中断的模块。

A.6.2 访问外设数据区

外设数据区可分为以下两部分：

- 用户数据
- 诊断与参数数据。

两个数据区都具有一个输入区(仅能被读取)和一个输出区(仅能被写入)。

用户数据

用户数据将使用输入或输出区的字节地址(对于数字信号模块)或单字地址(对于模拟信号模块)进行寻址。使用装载和传送命令、通讯功能(操作员接口访问)或通过传送过程映像，均可访问用户数据。用户数据可以是下面任意一种：

- 来自信号模块的数字和模拟输入/输出信号
- 来自功能模块的控制与状态信息
- 来自通讯模块的点对点和总线连接信息(仅适用于 S7-300)

在传送用户数据时，数据一致性最高可达到 4 个字节(DP 标准从站除外，参见 设置操作特性)。如果使用“传送双字”语句，则传送四个邻接且未修改的(相容的)字节。如果使用四个单独的“传送输入字节”语句，则硬件中断 OB 将被插入到语句之间，并将数据传送给同一地址，从而在传送原来这 4 个字节之前会对这 4 个字节的内容进行修改。

诊断与参数数据

模块的诊断与参数数据不能分开寻址，而是始终以完整数据记录的形式进行传送。这意味着将始终传送相容的诊断与参数数据。

使用模块的起始地址和数据记录号对诊断与参数数据进行访问。数据记录可分为输入和输出数据记录。输入数据记录只能被读取，而输出数据记录只能被写入。使用系统功能或通讯功能(用户接口)均能访问数据记录。

下表说明了数据记录与诊断和参数数据之间的联系。

数据	描述
诊断数据	如果模块具有诊断功能，则通过读取数据记录 0 和 1 获得模块的诊断数据。
参数数据	如果模块是可组态的，则通过写入数据记录 0 和 1 将参数传送给模块。

访问数据记录

可使用模块数据记录中的信息将参数重新分配给可组态的模块，并从具有诊断功能的模块中读取诊断信息。

下表列出了可用来访问数据记录的一些系统功能。

SFC	用途
将参数分配给模块	
SFC55 WR_PARM	将可修改的参数(数据记录)分配给所寻址的信号模块
SFC56 WR_DPARM	将 SDB 100 至 129 中的参数分配给所寻址的信号模块
SFC57 PARM_MOD	将 SDB 100 至 129 中的参数分配给所寻址的信号模块
SFC58 WR_REC	将任意数据记录分配给所寻址的信号模块
读出诊断信息	
SFC59 RD_REC	读取诊断数据

注意

如果使用一个 GSD 文件(版本 3 以上的 GSD)对 DPV1 从站进行了组态，且 DP 主站的 DP 接口已设置为“**S7 兼容**”，则数据记录将不能对带有 SFC 58/59 或 SFB 53/52 的用户程序中的 I/O 模块中进行读取或写入。

纠正方法：将 DP 主站的接口设置为“DPV1”。

寻址 S5 模块

可通过如下方法访问 S5 模块：

- 使用接口模块 IM 4632 将 S7400 连接到 SIMATIC S5 扩展机架
- 将某个 S5 模块插入到 S7400 中央机架的适配器箱中

关于如何使用 SIMATIC S7 对 S5 模块进行寻址，请参见“S7-400、M7-400 可编程控制器，硬件与安装”手册中的说明以及关于适配器箱的有关描述。

A.7 设置操作特性

A.7.1 设置操作特性

本章将说明如何通过设置系统参数或使用系统功能(SFC)来修改 S7-300 和 S7400 可编程控制器的某些属性。

关于模块参数的详细信息，请参见 STEP 7 在线帮助以及下列手册：

- “S7-300 可编程控制器，硬件与安装”手册
- “S7-300、M7-300 可编程控制器，模块技术规范说明”参考手册
- “S7-400、M7-400 可编程控制器，模块技术规范说明”参考手册

关于 SFC 的所有信息，可参见“S7-300 和 S7-400 的系统软件，系统功能与标准功能”参考手册。

寻址 DP 标准从站

如果要与 DP 标准从站交换 4 个以上字节的数据，则必须使用用于该数据交换的专用 SFC。

支持通过 I/O 区交换一致性数据(> 4 个字节)的 CPU 不需要使用 SFC 14/15(参见 一致性数据的分布式读取和写入)。

SFC	**用途**
将参数分配给模块	
SFC15 DPWR_DAT	将任意数据分配给所寻址的信号模块
读出诊断信息	
SFC13 DPNRM_DG	读取诊断信息(异步读访问)
SFC14 DPRD_DAT	读取一致性数据(长度为 3 个字节或大于 4 个字节)

当 DP 诊断帧到达时，具有 4 个字节诊断数据的诊断中断将给 CPU 发出信号。使用 SFC13 DPNRM_DG.可读出这 4 个字节。

A.7.2 改变模块的行为与属性

缺省设置

- 在提供了缺省设置时，S7 可编程控制器的所有可组态模块都具有适用于标准应用程序的缺省设置。有了这些缺省设置，不需进行任何设置就可立即使用这些模块。关于缺省设置的解释说明，请参见下列手册中的模块描述：
- “S7-300 可编程控制器，硬件与安装”手册
- “S7-300、M7-300 可编程控制器，模块技术规范说明”参考手册
- “S7-400、M7-400 可编程控制器，模块技术规范说明”参考手册

可为哪些模块分配参数？

可对模块的行为与属性进行修改，以满足您的要求，适应工厂环境。可组态的模块是 CPU、FM、CP 以及某些模拟输入/输出模块和数字输入模块。

可组态模块可带和不带备份电池。

不带备份电池的模块在出现任何电源故障后必须重新为其提供数据。这些模块的参数均存储在 CPU 的保留存储区中(由 CPU 间接分配参数)。

设置并装载参数

可使用 STEP 7 设置模块参数。在保存参数时，STEP 7 将创建对象“系统数据块”，该数据块将由用户程序下载给 CPU，并在 CPU 启动时传送给模块。

可进行哪些设置？

可将模块参数分为参数块。CPU 与可用参数块的对应关系，请参见“S7-300 可编程控制器，硬件和安装”手册和“S7-400、M7-400 可编程控制器，模块技术规范说明”参考手册。

参数块的实例：

- 启动特性
- 循环
- MPI
- 诊断
- 保留数据
- 时钟存储器
- 中断处理

- 板载 I/O(仅适用于 S7-300)
- 防护等级
- 本地数据
- 实时时钟
- 异步错误

使用 SFC 进行参数分配

除了可使用 STEP 7 进行参数分配外，也可在 S7 程序中加入系统功能，以修改模块参数。下表列出了哪些 SFC 可传送哪些模块参数。

SFC	用途
SFC55 WR_PARM	将可修改的参数(数据记录 1)传送给所寻址的信号模块
SFC56 WR_DPARM	将来自相应 SDB 中的参数传送给所寻址的信号模块
SFC57 PARM_MOD	将来自相应 SDB 中的所有参数传送给所寻址的信号模块
SFC58 WR_REC	将任意数据记录传送给所寻址的信号模块

关于系统功能的详细描述，请参见“S7-300 和 S7-400 的系统软件，系统功能和标准功能”参考手册。

关于可动态修改哪些模块参数，请参见下列手册：

- “S7-300 可编程控制器，硬件与安装”手册
- “S7-300、M7-300 可编程控制器，模块技术规范说明”参考手册
- “S7-400、M7-400 可编程控制器，模块技术规范说明”参考手册

A.7.3 离线更新模块和子模块中的(操作系统)固化程序

下文描述了如何通过存储卡将一个新的固件版本(=新操作系统版本)传送到模块或CPU。

要将更新文件传送给存储卡，请按如下执行：

1. 通过 Windows 浏览器创建一个新的目录。
2. 从更新软盘将 UPD 文件复制到该目录。
3. 在 SIMATIC 管理器中选择菜单命令 PLC > 更新操作系统。
4. 在所显示的对话框中选择具有 UPD 文件的目录。
5. 选择任意一个 UPD 文件。
6. 点击"确定"关闭对话框。

在可编程控制器中插入存储卡。

执行操作系统更新：

1. 切断插入了 CPU 的机架电源(PS)。
2. 将准备好的带操作系统更新程序的存储卡插入到 CPU 中。
3. 接通插有 CPU 的机架上的电源。
   操作系统从存储卡传送到内部 FLASH EPROM。
   此时，CPU 上的所有 LED 都点亮。
4. 约 2 分钟之后，更新完成。为了指示更新完成，CPU 上的 STOP LED 慢速闪烁(系统请求存储器复位)。
5. 切断电源单元的电源，适当时，插入用于该操作的存储卡。
6. 接通电源。CPU 自动执行存储器复位。之后，CPU 准备就绪，可进行操作。

A.7.4 使用时钟功能

所有 S7-300/S7-400 CPU 都配备有时钟(实时时钟或软件时钟)。在可编程控制器中，时钟既可作为外部同步的时钟控制方也可作为时钟从属方。时间中断和运行仪表都需要时钟。

时间格式

时钟始终指示时间(最小分辨率为 1 秒)、日期和星期几。也有一些 CPU 可以指示毫秒(参考“S7-300 可编程控制器，硬件和安装”手册和“S7-400、M7-400 可编程控制器模块规范”参考手册)。

设置和读取时间

可以调用用户程序中的 SFC0 SET_CLK 来设置 CPU 时钟的时间和日期，或使用编程设备的菜单选项来启动时钟。使用 SFC1 READ_CLK 或编程设备的菜单选项，可以读取 CPU 的当前日期和时间。

注意

为了防止在不同的 HMI 系统上时间显示不同，应当将 CPU 设置为**冬令时**。

分配时钟参数

如果在网络中，装备了不止一个具有时钟的模块，必须使用 STEP 7 设置参数，来指定在时间同步时哪个 CPU 功能是主控的，哪个是从属的。当设置这些参数时，还要决定是通过通讯总线还是通过多点接口来同步时间，以及自动时间同步的间隔。

同步时间

要确保网络中所有模块的时间相同，从属的时钟将以固定(可选择的)的间隔由系统程序进行同步。使用系统功能 SFC48 SFC_RTCB，可以将控制时钟的日期和时间传送到从属时钟。

使用运行仪表

运行仪表计算已连接设备的运行小时数或 CPU 总的运行小时数。

在 STOP 模式时，运行计时表也将停止。即使存储器复位后，它的计数值还能保持。在重新启动(暖启动)时，运行计时表必须由用户程序重启；在热启动时，如果原先已经将其启动，它会自动继续。

可以使用 SFC2 SET_RTM 设置运行计时表的初始值。可以使用 SFC3 CTRL_RTM 启动或停止运行计时表。可以使用 SFC4 READ_RTM 读取当前总的运行小时数和计数器的状态(“已停止”或“正在计数”)。

一个 CPU 可以拥有多达八个运行计时表。编号从 0 开始。

A.7.5 使用时钟存储器和计时器

时钟存储器

时钟存储器是一个可以按照 1:1 的脉冲-暂停比率周期性地改变其二进制状态的存储器字节。当使用 STEP 7 为时钟存储器分配参数时，可选择在 CPU 中使用哪个存储器字节。

使用

例如，您可在用户程序中使用时钟存储器字节来激活闪烁的灯光或触发周期性动作(例如测量实际值)。

可用频率

时钟存储器字节的每一位均分配了一个频率。下表显示了这种分配：

时钟存储器字节的位	7	6	5	4	3	2	1	0
持续周期(秒)	2.0	1.6	1.0	0.8	0.5	0.4	0.2	0.1
频率(赫兹)	0.5	0.625	1	1.25	2	2.5	5	10

注意

时钟存储器字节与 CPU 周期并不同步，即在较长周期内，时钟存储器字节的状态将改变好几次。

定时器

计时器是系统存储器中的一块存储区。您可在用户程序中设定计时器的功能(例如，时延计时器)。可供使用的计时器数量取决于 CPU。

注意

如果在用户程序中使用的计时器个数超过 CPU 允许，则将发出一个同步错误信号，并启动 OB121。

在 S7-300 上(除 CPU 318 外)，计时器只能在 OB1 和 OB100 中同时启动和更新；而在所有其它 OB 中，则只能启动计时器。

索引

字母

A

B

C

D

H

J

K

T

W

X